AFRL-IF-RS-TR-2005-209
Final Technical Report
May 2005

ROBUST LOSSLESS IMAGE DATA HIDING

New Jersey Institute of Technology

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

STINFO FINAL REPORT

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2005-209 has been reviewed and is approved for publication

APPROVED: /s/

CHAD D. HEITZENRATER
Project Engineer

FOR THE DIRECTOR: /s/

JOSEPH CAMERA, Chief
Information & Intelligence Exploitation Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	MAY 2005	Final Sep 03 – Sep 04

4. TITLE AND SUBTITLE
ROBUST LOSSLESS IMAGE DATA HIDING

5. FUNDING NUMBERS
C - F30602-03-1-0264
PE - 63789F
PR - STG3
TA - 03
WU - 10

6. AUTHOR(S)
Yun Q. Shi

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
New Jersey Institute of Technology
Department of Electrical & Computer Engineering
323 M. L. King Boulevard
Newark New Jersey 07102

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Research Laboratory/IFEC
525 Brooks Road
Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2005-209

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Chad D. Heitzenrater/IFEC/(315) 330-2575/ Chad.Heitzenrater@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

Two novel robust lossless data hiding algorithms that do not use modulo-256 to achieve losslessness and hence do not generate salt-and-pepper noise have been invented and developed in this project. One is implemented in spatial domain, another in integer wavelet transform domain. Both are based on patchwork theory and both treat blocks of different content with different embedding schemes. Permutation and error correction codes are utilized to enhance robustness. Extensive experimental works have been conducted and have demonstrated the effectiveness of these two algorithms. These two algorithms have been used in a Unified Authentication Framework for JPEG2000 Images, which has now been included in the final committee draft (FCD) 1.0 of the Security Part of JPEG2000 (JPSEC) standard.

14. SUBJECT TERMS
Watermarking, Lossless Data Hiding, Reversible Data Hiding, Robust Lossless Data Hiding, Semi-Fragile Authentication

15. NUMBER OF PAGES
63

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

Summary

Recently a new subset of data hiding techniques, lossless data hiding, has received increased interest. Most of the existing lossless data hiding algorithms are, however, fragile in the sense that the hidden data cannot be extracted correctly after compression or other incidental (or non-incidental) alterations have been applied to the stego-image. If the hidden data can still be extracted correctly after the alterations have been applied, the lossless data hiding algorithm is called semi-fragile (referred to as robust in this paper). The only existing robust lossless data hiding technique that is robust against high-quality JPEG compression is based on modulo-256 addition to achieve its losslessness. This technique suffers from salt-and-pepper noise caused by the use of modulo-256 addition to prevent overflow/underflow.

In this report, two novel robust lossless data hiding techniques are proposed, one implemented in the spatial domain and another in integer wavelet domain. Both approaches do not generate salt-and-pepper noise because they do not use modulao-256 addition to gain losslessness. The spatial domain technique, based on the patchwork theory, uses a robust statistical quantity for embedding. Error correction codes (ECC) and a permutation scheme is utilized and differentiating the bit-embedding processes based on the pixel group's content are employed to achieve both losslessness and robustness. For the technique implemented in the integer wavelet domain, a robust quantity in the integer wavelet domain has been identified for data embedding. Special measures are taken based on integer wavelet transform theory so that the bit-embedding process is differentiated according to the block content of selected integer wavelet coefficients. Similarly, error correction coding and permutation are also used to achieve losslessness. Both techniques have been successfully applied to many images, thus demonstrating their generality. The experimental results show that the high visual quality of stego-images, the data embedding capacity, and the robustness of the proposed lossless data hiding schemes against compression are acceptable for many applications, including semi-fragile image authentication. Specifically, they have been successfully applied to authenticate losslessly compressed JPEG2000 images, followed by possible transcoding. It is

expected that these two new robust lossless data hiding algorithms can be readily applied to the medical field, law enforcement, remote sensing, and other areas where the recovery of original images is desired.

1. Introduction

In data hiding, information (represented as binary data) is hidden in a cover media. This is done by changing the cover media at the constructive level. Data hiding has been suggested as a promising technique for the purposes of authentication, fingerprinting, security, data mining, and copyright protection.

Many image data hiding algorithms have been proposed in the past several years. In most cases, the cover media will experience some permanent distortion due to data hiding, which cannot be inverted to restore the original cover [1-5]. In some applications, such as in the fields of law enforcement and medical imaging, in addition to perceptual transparency it is desirable, for legal reasons, to reverse the marked media back to the original cover media without any distortions after the hidden data is retrieved. Other examples where the ability to reverse data embedding distortion is desired include military imaging systems and remote sensing, where high precision is required; and in some scientific research such as high-energy physics, where experiments are expensive. The marking techniques satisfying this reversibility requirement are referred to as reversible, lossless, distortion-free, or invertible data hiding techniques. According to our survey [6], many lossless data hiding techniques have been proposed, such as those reported in [7-14]. However, most of them are fragile in the sense that the hidden data cannot be recovered after compression or other incidental alteration has been applied to the marked image. Thus far, De Vleeschouwer et al.'s method [14] is the only lossless data hiding technique robust to high-quality JPEG compression.

De Vleeschouwer et al.'s method can be applied to semi-fragile image authentication; that is, in the case that the marked image does not change at all, the hidden data can be extracted correctly and the original image can be recovered losslessly, hence the image is rendered authentic. In the case that the marked image goes through compression to some extent, the hidden data can still be correctly extracted for semi-fragile authentication if the hidden data represents the compressed version of the image. This is useful, as semi-fragile authentication may be more practical than fragile authentication for many applications since it allows some incidental modification (for instance, image compression) through which the perceived content of the image has not been changed.

The main idea of De Vleeschouwer et al.'s method is based on the patchwork theory [15] and modulo-256 addition. Below is a brief overview of the algorithm.

1. First, each bit of the hidden data is associated with a group of pixels, e.g., a block in an image. Each block is randomly divided into two sets of pixels with an equal size, named zones A and B. The histogram of each zone is mapped to a circle (Figure 1) where positions on the circle are indexed by the corresponding grayscale values, and the weight of a position is the number of pixels assuming the corresponding grayscale value.

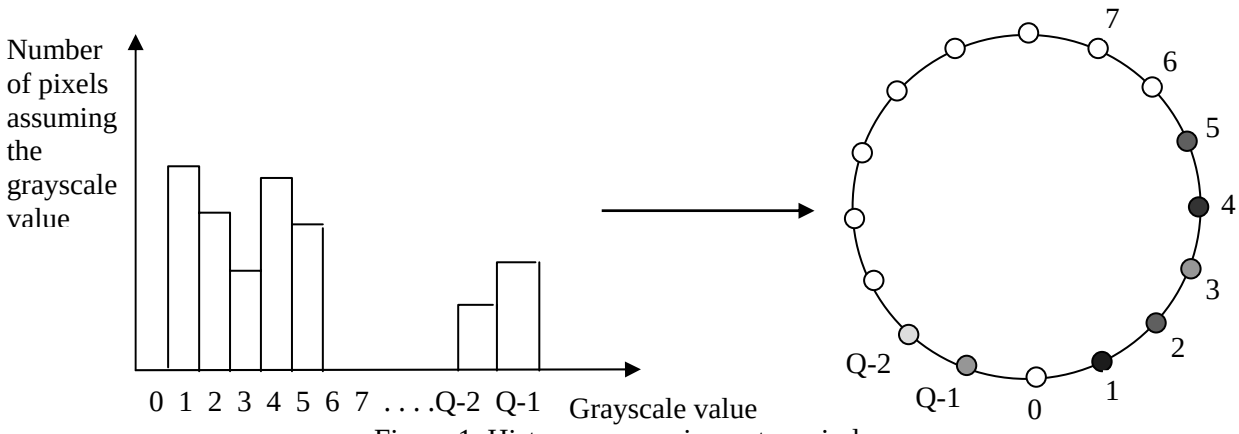


Figure 1. Histogram mapping onto a circle.

2. In Figure 2, vectors C_a and C_b point from the center of the circle to the *mass* center of zones A and B, respectively. Note that the mass center is calculated from the grayscale values and their weights described in the above point. Since zones A and B are two pseudo-randomly divided sets of equal size within the same block, it is highly probable that vectors C_a and C_b are similar to each other. Slight rotation of these two vectors in opposite directions allows embedding a bit of information in the block. For example, C_a can be rotated clockwise and C_b counter-clockwise to embed a bit '0', while C_a is rotated counter-clockwise and C_b clockwise to embed a bit '1'. As to the pixel grayscale values, the rotation of these vectors is equivalent to shifting of the associated grayscale values by a corresponding amount.

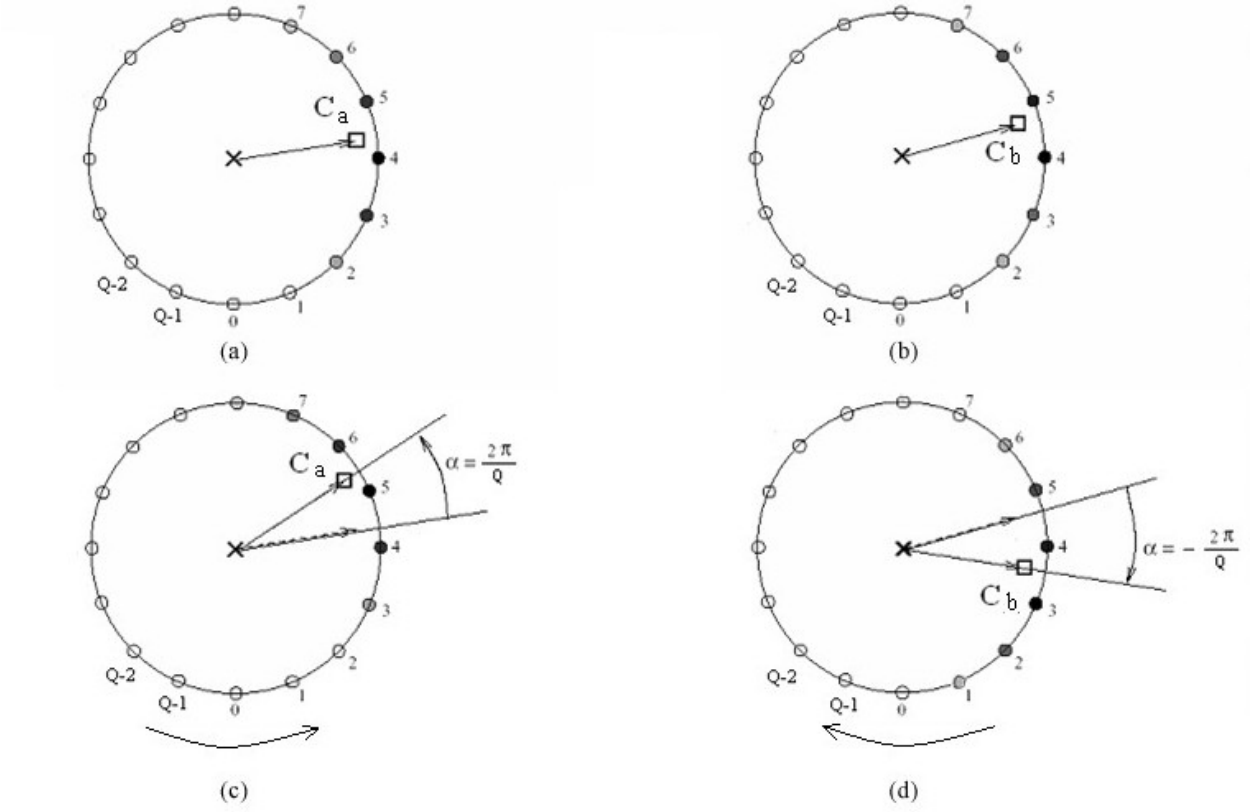


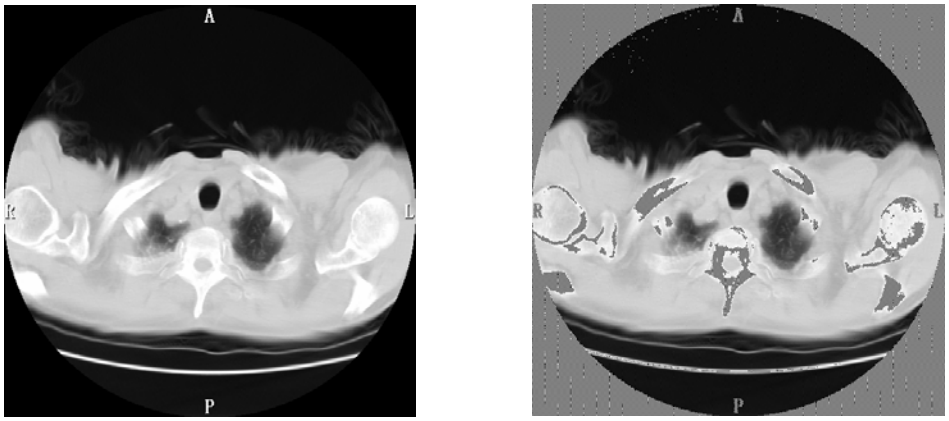
Figure 2. Data (bit "1") embedding diagram.

3. The data extraction process is the inverse process of the data embedding. The extraction process first partitions the image into blocks, and then into zones A and B in the same way as that used in the embedding process. The histogram of each zone is then mapped onto the corresponding circle, and the center of mass is computed for both, again as in embedding. Let V denote the difference of the orientation angles between the vectors C_a and C_b. The sign of V provides information as to the rotation directions during the embedding process and hence enables bit retrieval. After data extraction, C_a and C_b can be rotated back to the original position, thus achieving reversibility.

The angle difference between the vectors of C_a and C_b depends on all of the pixel grayscale values in zones A and B, respectively. From patchwork theory, it can be seen that this method is possibly robust against high-quality JPEG compression. Experimental results have verified this claim. However, our extensive investigation has revealed a severe problem suffered by this technique.

A common problem in data embedding is the overflow/underflow problem. Overflow and underflow occur when, after data embedding, the grayscale values of some pixels in the marked image exceed the upper bound (255 for a grayscale image having eight-bits per pixel) and/or the lower bound (0 for eight-bit grayscale images). This situation necessitates the use of truncation, hence violating the principle of lossless data hiding. Therefore, avoiding the overflow/underflow problem is a key issue in lossless data hiding. From Figure 2, it is noted that modulo-256 addition is used in this method to handle the overflow/underflow problem for achieving losslessness. Therefore, this algorithm generates "salt-and-pepper noise". Salt and pepper noise occurs when, in doing modulo-256 addition, a very bright pixel with a large grayscale value close to 255 is changed to a very dark pixel with a small grayscale value close to 0, and vice versa. One example is shown in Figure 3, where De Vleeschouwer et al.'s algorithm is applied to a medical image, Mpic1. Severe salt-and-pepper noise can be observed, so much so that the name "*salt-and-pepper*" becomes improper. Medical images often contain many rather dark and/or rather bright pixels, and therefore have high potential for suffering from severe salt-and-pepper noise. Salt-and-pepper noise may also be severe for daily-life images as well. Figure 4 presents an example of a rather severe

salt-and-pepper noise case on a color image, Woman (one of the JPEG2000 test images). In this example the algorithm has been applied to the Red component of the image. The salt-and-pepper noise manifests itself as severe color distortion, as half of her hair area has changed from black to red, while the color of most of her right-hand palm area has changed from flesh tone to green. Note that authors of [14] also proposed an optional method in the same paper to overcome the salt-and-pepper noise. However, as stated in their paper, this new method is a fragile, instead of semi-fragile, lossless data hiding method.



(a) Original (b) Marked

Figure 3. A medical image, Mpic1.

(a) Original (b) Marked

Figure 4. A JPEG2000 test image, Woman. (Note that there are some regions in the stego-image which have experienced color distortion and have been circled with white lines.)

Another drawback to the De Vleeschouwer, et al. scheme is that the marked image does not have high PSNR with respect to the original image. Table 1 and Table 2 summarize the performance of [14] applied to eight medical images and eight JPEG2000 test images. Note that *block size* refers to the number of pixels along one side of a square block, while *embedding level* represents the amount of grayscale value change for each pixel in the block when either bit '1' or '0' is embedded into the block. Also note that the embedding capacity is different among eight JPEG2000 test images, due to the use of different Bose-Chadhuri-Hocquenghem (BCH) error-correction codes. For instance, for those images with a capacity of 1410 bits, BCH (31,6) is used, while for those images with a capacity of 805 bits BCH (63,7) is used. The BCH codes and the motivation for using BCH codes will be discussed later in this report. It is observed from Table 1 that when 100 information bits are embedded in eight 512×512 medical images the PSNR is below 30 dB. Five marked images suffer from severe salt-and-pepper noise with a PSNR of only 10 dB or below. In Table 2, when 805 or 1410 bits are embedded in eight 1536×1920 JPEG2000 color test images, the PSNR is below 25 dB. Five marked images suffer from severe salt-and-pepper noise resulting in a PSNR below 20 dB. Note that in Tables 1 and 2, *robustness* indicates the surviving bit rate in the unit of bpp (bits per pixel), i.e., when a compressed image has its resultant data rate above or equal to this bit rate, the hidden data can be retrieved without error.

Table 1. Test results for eight medical images with block size as 8, embedding level as 6.

Images (512x512)	PSNR of marked image (dB)	Data embedding capacity (bits)	Robustness (bpp)
Mpic1	9.93	100	0.8
Mpic2	4.94	100	1.6
Mpic3	28.10	100	0.8
Mpic4	28.21	100	0.4
Mpic5	28.21	100	0.8
Mpic6	5.86	100	2.0
Mpic7	10.52	100	0.8
Mpic8	6.26	100	1.6

Table 2. Test results for eight JPEG2000 test images with block size as 20, embedding level as 8.

Images (1536x1920)	PSNR of marked image (dB)	Data embedding capacity (bits)	Robustness (bpp)
N1A (Woman)	17.73	1410	0.8
N2A	17.73	1410	2.2
N3A	23.73	1410	0.6
N4A	19.67	1410	1.2
N5A	17.28	1410	1.2
N6A	23.99	805	0.6
N7A	20.66	1410	1.4
N8A	14.32	805	1.4

Therefore, as can be seen from the above experimental results, the observation has been made that all of the lossless data hiding algorithms based on modulo-256 addition (e.g., [7,14]) are not acceptable for many practical uses. Thus, a new robust, lossless data hiding technology that does not use modulo-256 addition (and therefore avoids the above-mentioned drawbacks) is called for.

2. A Novel Robust Lossless Image Data Hiding Algorithm Implemented in Spatial Domain

In order to develop an embedding scheme that is robust against image compression, it is necessary to select a robust parameter and manipulate it to embed data. The statistical quantity selected as the parameter for the proposed scheme will now be discussed.

2.1. A robust statistical quantity used to embed data

For a given image block (for example, 8 pixels $\times$ 8 pixels), the block is split into two sets A and B as shown in Figure 5. Set A consists of all pixels marked by '+', denoted by a_i, and the set B is marked as '-', denoted as b_i. Each set has 32 pixels. According to patchwork theory, different ways to split sets A and B are possible; however, the splitting method described will be used in this report.

+	-	+	-	+	-	+	-
-	+	-	+	-	+	-	+
+	-	+	-	+	-	+	-
-	+	-	+	-	+	-	+
+	-	+	-	+	-	+	-
-	+	-	+	-	+	-	+
+	-	+	-	+	-	+	-
-	+	-	+	-	+	-	+

Figure 5. Difference pair pattern.

Note that it is not required that the block size be 8×8. As will be demonstrated, each block is used to embed one bit. Therefore, the block size will affect data embedding capacity; the larger the block size, the lower the data embedding capacity. The robustness of embedded bits, on the other hand, will be stronger if the block size is larger. A compromise between the data embedding capacity and robustness of hidden data needs to be made according to specific applications.

For each block a difference value α is calculated, which is defined as the arithmetic average of the differences in the grayscale values of pixel pairs within the block. A pixel pair is chosen (in this example, we will use two horizontally neighboring pixels; one marked as '+',and the other '-', with each pixel used only once) and α is calculated according to the formula below, where n is the number of pixel pairs in the block. In this example, n is equal to 32.

$$\alpha = \frac{1}{n} \sum_{i=1}^{n} (a_i - b_i)$$

Since the pixel grayscale values in a local block are often highly correlated and have spatial redundancy, the difference value α is expected to be very close to zero. The experimental results have supported this observation. For example, the distribution of the difference value α of blocks in the 'Boat' image is shown in Figure 6. One can see from this figure that most values of α are very close to zero (in other words, the mean value of this distribution is zero). The α distributions of other images also follow this

pattern. Many other strategies to split sets A and B have been investigated; however, the splitting strategy shown in Figure 5 seems to have the least variance of α values and therefore results in the least visual distortion in the marked images versus the original image.

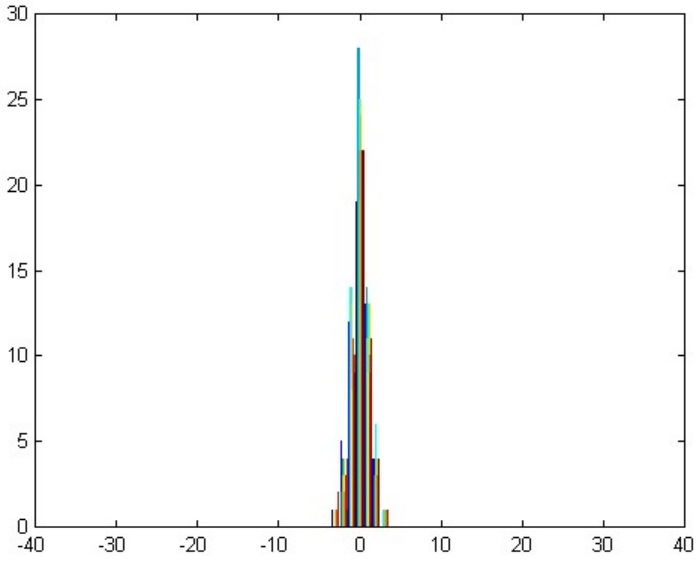


Figure 6. The distribution of difference value α.

Since the difference value α is based on the statistics of all pixels in the block, it has a certain amount of robustness against JPEG/JPEG2000 compression and other small, incidental alterations. For this reason, this value will be used as the robust quantity for data embedding.

2.2. Differentiating bit-embedding schemes based on different grayscale distributions within a block of pixels

With the cover image divided into non-overlapping blocks, one bit is embedded in each block. In performing bit-embedding, the difference value α is kept within specified thresholds K and –K to embed bit '0' while the difference value α is shifted beyond the threshold K or –K to embed bit '1'. In the experiments conducted, K is generally kept to less than five. As analyzed in Section 1, although modulo-256 addition can avoid the overflow/underflow problem this approach has not been used as it will lead to salt-and-pepper noise. In order to overcome the overflow/underflow problem, the approach taken

is to classify the blocks into four different categories and use different bit-embedding schemes for each category. Theory and experimental results show that this technique successfully solves the overflow/underflow problem and avoids the salt-and-pepper noise at the same time.

In the proposed algorithm, a shift quantity (also referred to as embedding level) β is employed. In the experiments conducted, the value used for this variable was twice the value of the specified threshold K. Note that shifting α towards the right-hand side (refer to Figure 8) means adding a fixed shift quantity, β, to the grayscale value of each pixel marked by '+' in set A. Likewise, shifting α towards the left-hand side (refer to Figure 8) means subtracting a fixed shift number, β, from the grayscale value of each pixel marked by '+' in set A. In the whole bit-embedding process, the grayscale values of pixels marked by '-' in set B are kept intact, thus reducing the distortion caused by the bit-embedding. Since error bits may be introduced in data embedding, error correction coding (ECC) is applied to correct them. Listed below are the detailed bit-embedding steps.

Category 1: The pixel grayscale values of a block under consideration are far enough away from the two bounds of the histogram (0 and 255 for an 8-bit grayscale image). That is, the distance $d = \min(d_l, d_r)$ satisfies $d \geq \beta$ (where β is the shift quantity), as shown in Figure 7.

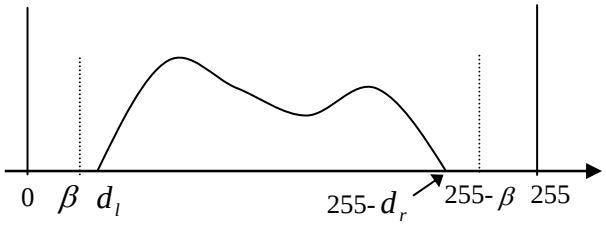


Figure 7. Block histogram for category 1.

In this category, we further consider the following two cases according to the value of α.

Case 1. The difference value α is located between the thresholds K and -K.

1. If the to-be-embedded bit is ‘1’, we shift the difference value α by a quantity β towards the right-hand side or left-hand side, depending on if α is positive or negative. Refer to Figure 8.

2. If to the to-be-embedded bit is ‘0’, the pixel value of that block is intact.

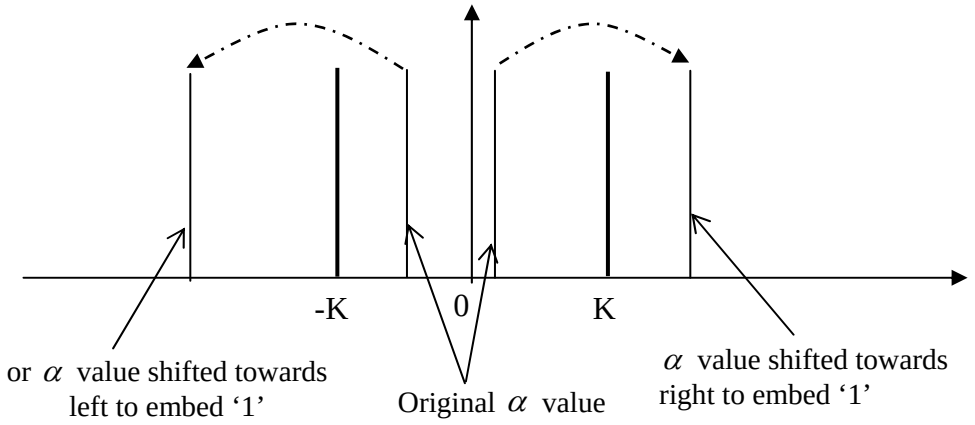


Figure 8. Embedding a bit ‘1’.

Case 2. The absolute value of α exceeds the threshold K.

In order to keep the data hiding lossless, no matter whether the to-be-embedded bit is ‘0’ or ‘1’ a bit value of ‘1’ is always embedded by shifting the difference value α further away from 0 by a quantity β, as shown in Figure 9. While an error bit may be introduced, it can be corrected by using ECC later.

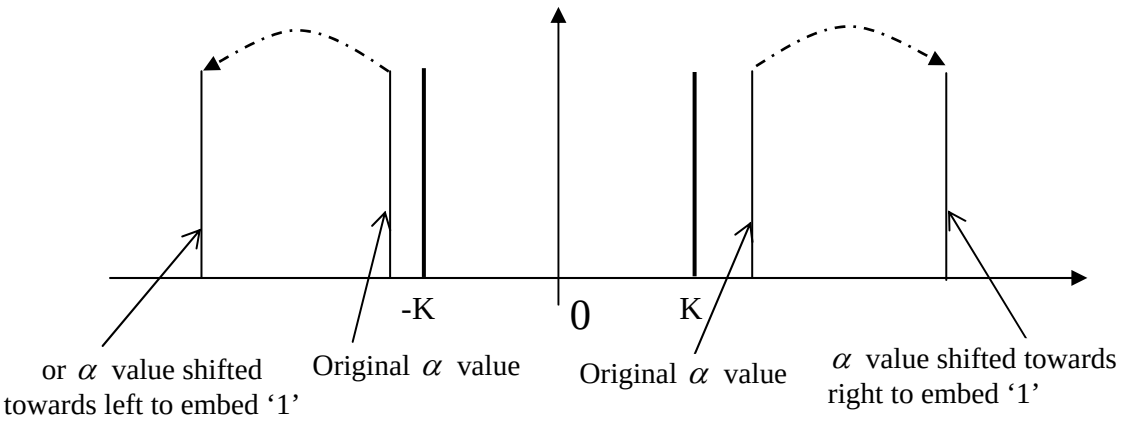


Figure 9. Embedding a bit ‘1’.

Category 2: Some pixel grayscale values of the block under consideration are very close to the lower bound of the histogram, i.e., 0 for an 8-bit grayscale image, while no pixel grayscale values are close to the upper bound of the histogram. This is depicted in Figure 10.

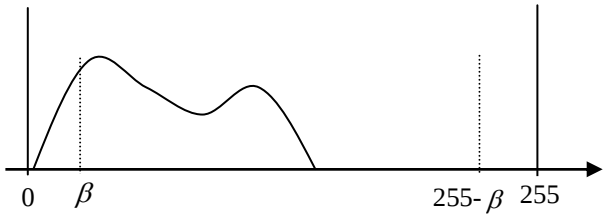


Figure 10. Block histogram for category 2.

In this category, we further consider three different cases according to the value α.

Case 1. The value α is located between the thresholds K and -K.

1. If the to-be-embedded bit is '1', we always shift the difference value α by a quantity β towards the right-hand side, beyond the threshold K. Refer to Figure 11.
2. If the to-be-embedded bit is '0', the pixel value of that block is left intact.

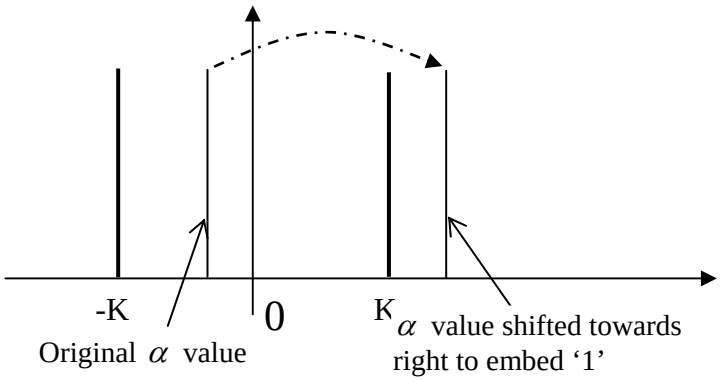


Figure 11. Embedding a bit '1'.

Case 2. The value α is located on the right-hand side, beyond the threshold K.

No matter whether the to-be-embedded bit is '0' or '1', a bit value of '1' is always embedded by shifting the difference value α by a quantity β, and thus making α further away from the zero

point, as shown in Figure 12. While an error bit may be introduced, it can be corrected by using ECC later..

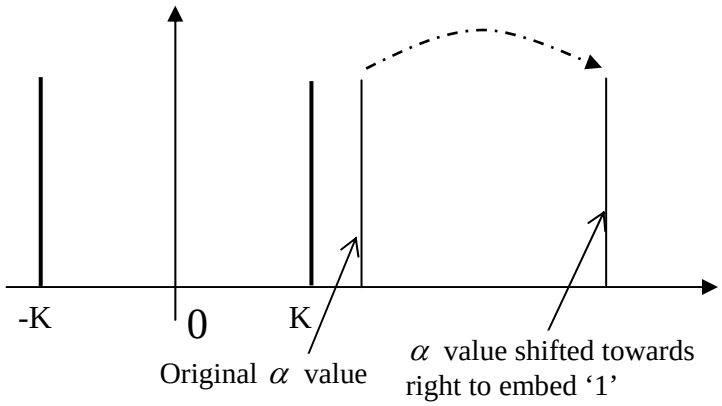


Figure 12. Embedding a bit '1'.

Case 3. The value α is located on the left-hand side, beyond the threshold -K.

In this case, two options are availible. Either the threshold value of K can be increased, or the block size can be enlarged until the α value becomes lager than –K. In this way, Case 3 becomes Case 1. With the newly selected value of K (or change in block size), the bit-embedding process is re-run block-by-block from the beginning. Over numerous experiments, it has been found that Case 3 is very rare. Furtehrmore, if it does happen, a slight increase of the K value or block size will work out.

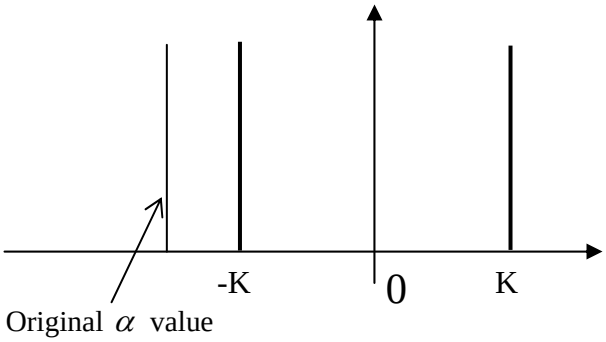


Figure 13. Increase threshold or block size to let α value greater than -K.

Category 3: Some pixel grayscale values of the block under consideration are very close to the upper bound of the histogram, while no pixel grayscale values are close to the lower bound of the histogram. This is depicted in Figure 14.

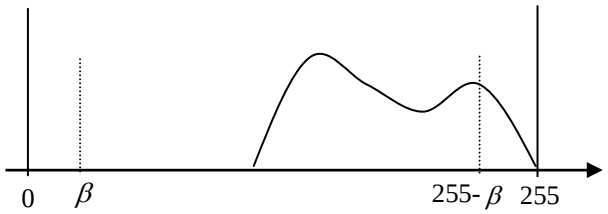


Figure 14. Block histogram for category 3.

Category 3 is similar to Category 2 except that the distribution of grayscale values of the block is close to the upper bound instead of the lower bound of the histogram. Hence, the data embedding process for Category 3 can be handled similar that of Category 2, except that when shifting the difference value α by a quantity β the shift will be to the left-hand side instead of to the right-hand side.

Category 4: Some pixel grayscale values of the block under consideration are close to the upper bounds, while some pixel grayscale values are close to the lower bounds of the histogram. This is depicted in Figure 15.

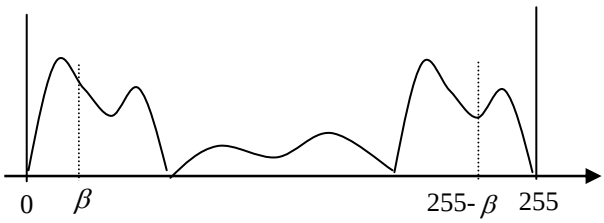


Figure 15. Block histogram for category 4.

In this category, we further consider two different cases according to the α value.

Case 1. The value α is located between the thresholds K and -K.

No matter whether the to-be-embedded bit is ‘0’ or ‘1’ a bit value ‘0’ is always embedded by keeping the difference value α intact, as shown in Figure 16. While an error bit may be introduced, it can be corrected by using ECC later

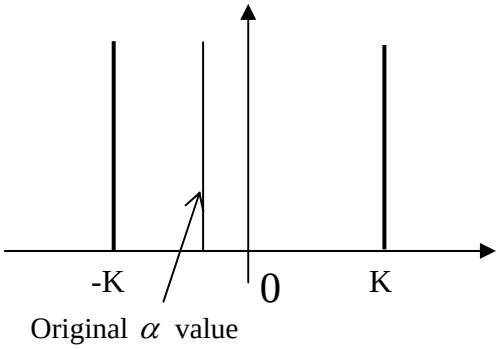


Figure 16. Embedding a bit ‘0’.

Case 2. The absolute value α is beyond the threshold K.

In this case, the pixel grayscale values for the block are not changed. A constant value is always embeded into the block (for instance, bit ‘0’) regadrless of the value of the the to-be-embedded bit. In decoding, the grayscale value distribution of the block is first examined. Once Case 2 of Category 4 is identified, bit ‘0’ is extracted, and the grayscale values of this block will remain unchanged. The possibly introduced error bit will be corrected using ECC.

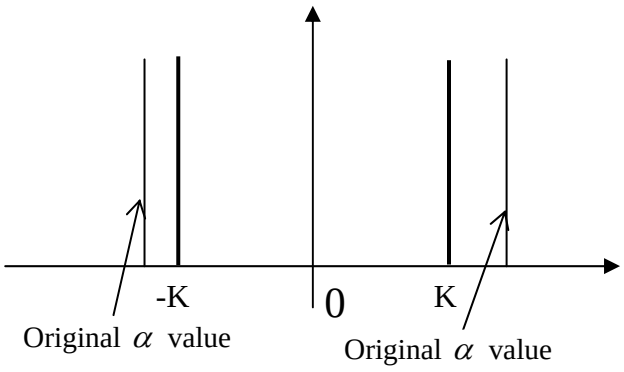


Figure 17. The pixel grayscale value of the block is unchanged. Bit ‘0’ is assumed to be embedded.

The above-mentioned four categories cover all situations that a block may encounter. The detailed description of the bit-embedding procedure demonstrates that the modified pixel grayscale value is still in the range of [0,255], and hence no overflow or underflow will take place.

It is noted that after data embedding some error bits will be generated. They will be handled in the following manner.

2.3. Error correction code

In the above-mentioned bit-embedding process, it was demonstrated that some instances might cause the embedder to introduce error bits. In order to recover the information bits correctly, an error correction code (ECC) is utilized to correct these error bits, at the price of sacrificing some data embedding capacity. Bose-Chadhuri-Hocquenghem (BCH) codes are a powerful class of cyclic codes that provide a large selection of block lengths, code rates, alphabet sizes, and error-correcting capability [16]. The proposed algorithm utilizes a set of BCH codes for selection; specifically, the codes BCH (15,11,1), BCH (15,7,2), BCH (15,5,3), BCH (31,6,7) and BCH (63,7,15) are used, respectively.

Note that there are three numbers within the parentheses after BCH, which denote the parameters of the code. The first number indicates how many bits in one codeword, the second number indicates the number of bits in one information symbol before ECC, and the third number represents how many bits can be corrected in one such codeword. The utilization of these BCH codes can then facilitate the trade-off between the coding ratio of ECC (hence the payload) and the error-correcting capability of ECC (hence robustness of lossless data hiding). For example, the BCH (63,7,15) code is the most powerful code among the listed codes in terms of error correction capability. It can correct 15 random error bits within a codeword of 63 bits. This of course comes at the price of more redundant bits, resulting in the smallest payload among these codes.

2.4. Permutation of message bits

For some images error bits may be concentrated in small areas in an image, which can possibly lead to too many error bits in one codeword and thus cause errors in data extraction. In this case, even the powerful

BCH (63,7,15) code cannot correct all of the error bits. To efficiently combat these *bursts of errors*, which may cause a failure in the algorithm, some kind of permutation scheme should be used together with ECC. As pointed out in [17], combining ECC and permutation is an effective and efficient strategy to combat both random errors and bursts of errors. For the sake of security, in the proposed algorithm the message bits are permuted using a secret key. In these experiments the chaotic mixing scheme described in [18] was used for the permutation.

2.6 Block diagram of proposed embedding scheme

Using the above-mentioned techniques, a data embedding diagram can be constructed as in Figure 18.

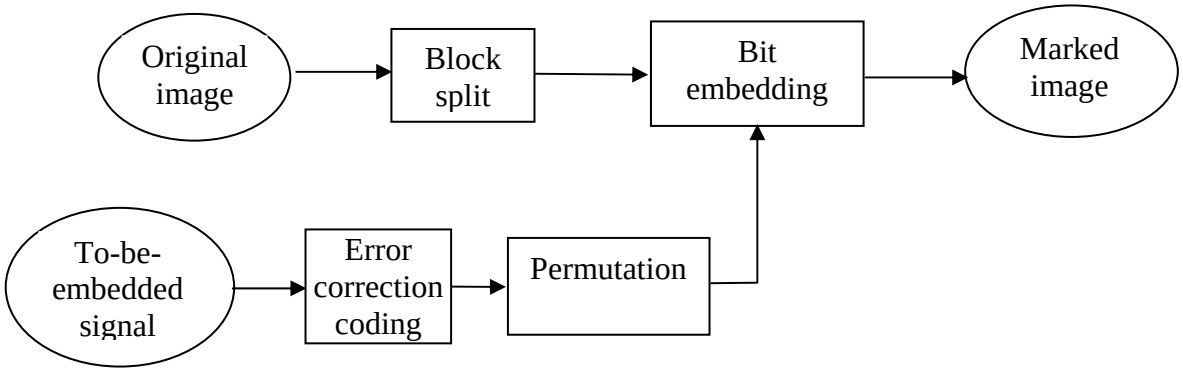


Figure 18. Block diagram of data embedding.

2.7 Data extraction

Data extraction is the reverse process of data embedding and is much simpler.

1. Given marked image, it is first split it into non-overlapping blocks of the same size as was used for embedding. The difference value α is then calculated for each block in the same way as that used in the embedding process.

2. If the absolute difference value α is larger than the threshold K, the grayscale value distribution of the block is examined. If the block is identified as Case 2 in Category 4, the bit '0' is extracted and the block remains unchanged. Otherwise, bit '1' is extracted and the difference value α is shifted back towards the zero point by adding or subtracting the quantity β, depending on whether the difference

value α is negative or positive. In this way, the difference value α is reverted back to its original value, which means each pixel grayscale value in set A is reset to its original value. This process is shown in Figure 19.

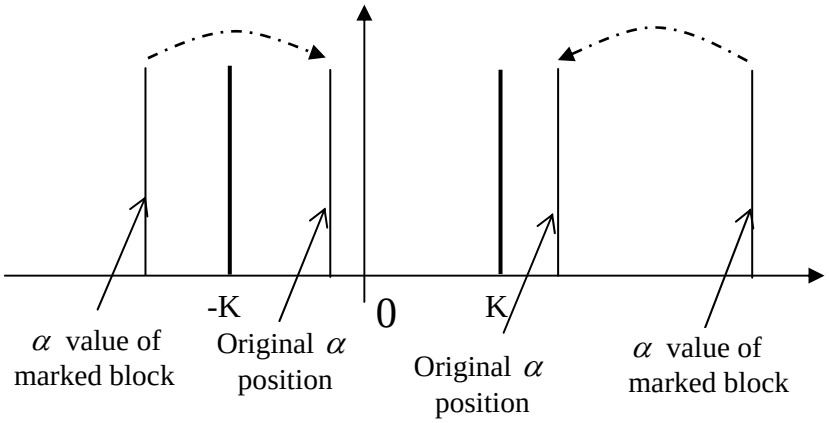


Figure 19. Extracting bit '1'.

3. If the absolute value of the difference value α is less than the threshold K, then bit '0' is extracted and nothing is done to the pixel grayscale value of that block.

Note that by combining this step and the above step in data extraction, it is obvious that all pixel grayscale values will be the same as in the original image.

4. After data extraction, the inverse permutation and the ECC decoding are applied, respectively, so as to obtain the original information bits correctly.

In this way, the original information bits can be extracted and the original image recovered without any distortion. The whole data extraction diagram is depicted in Figure 20.

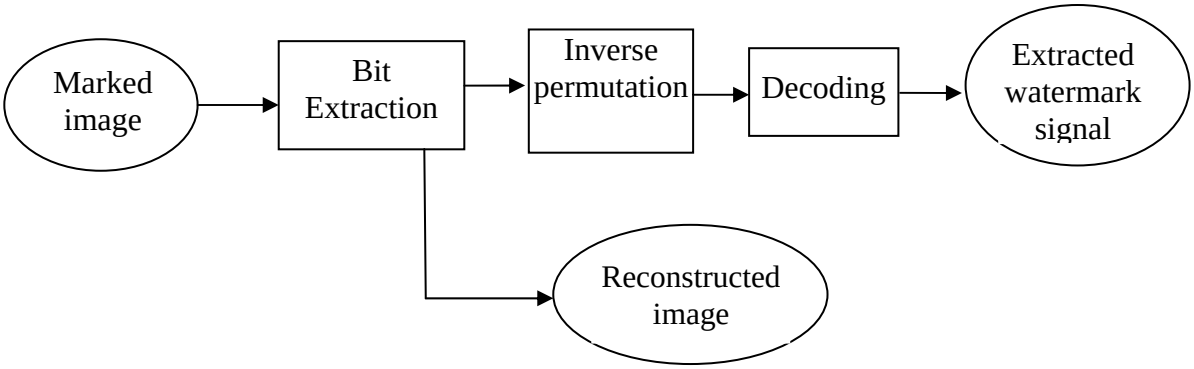


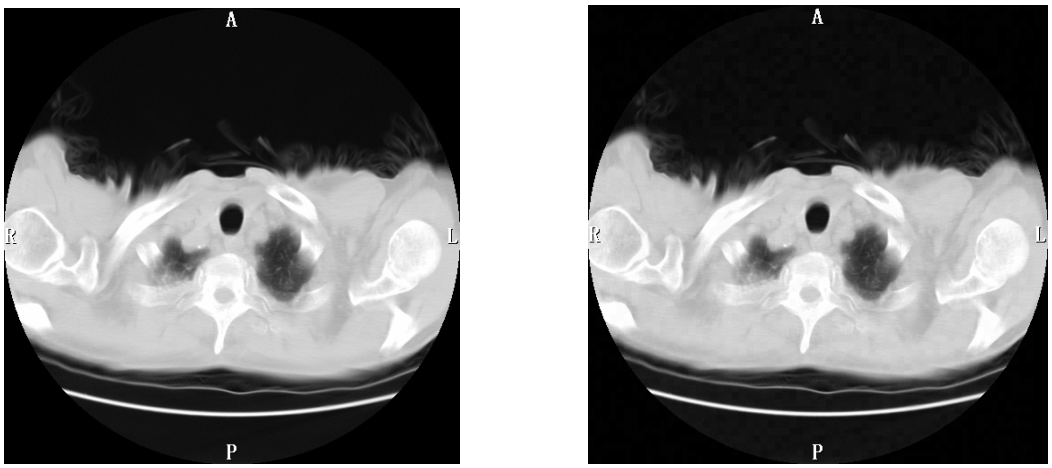
Figure 20. Block diagram of data extraction.

2.8 Experimental Results

The proposed algorithm has successfully been applied to commonly used grayscale images ('Lena', 'Baboon', 'Boat'), eight medical images, eight JPEG2000 color test images, and all 1096 images in the CorelDraw image database. For color images, the algorithm is applied only to one color plane, and the program has been written to automatically select and embed data into color plane that results in the best performance. Note that there is no salt-and-pepper noise in all of the tests completed. The embedding capacity from these experiments ranged from 512 to 1024 bits for the purpose of authentication, and can be adjusted for other applications by changing the block size. As shown in tables later in this section, the PSNR is much higher than that obtained by using the method in [14]. It is noted that the data embedding capacity and the PSNR of the marked image versus the original image can be adjusted according to the requirements by adjusting the parameters. Since these two performance parameters are usually in conflict with each other (in the sense that if the embedding capacity is improved, the PSNR will drop and vice versa), there is usually a trade-off between the data embedding capacity and the PSNR of the marked image versus the original image for a targeted application.

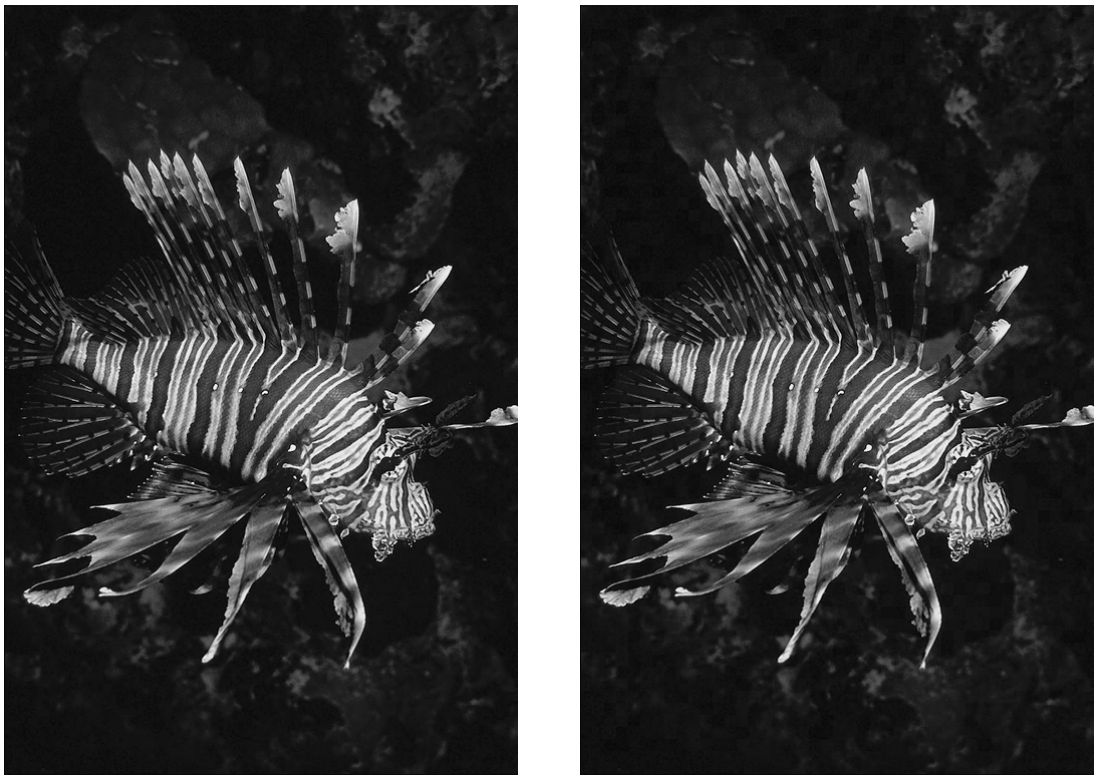
The tested images have been shown to resist JPEG/JPEG2000 compression, with the surviving bit rate ranging from 2.0 bpp (bits per pixel) to 0.2 bpp (bits per pixel). In other words, the hidden data can be retrieved with no errors when image compression is applied to marked images with the resultant bit rate equal to or greater than the above-mentioned surviving bit rate.

The following are some example test images. Note that no visible artifacts exist, indicating that a significant performance improvement has been achieved as compared with [14]. Also, there is no color distortion at all in the marked Woman image, as shown in Figure 23 (b).



(a) Original (b) Marked

Figure 21. A medical image, Mpic1.



(a) Original (b) Marked

Figure 22. A CorelDraw image.

(a) Original (b) Marked

Figure 23. A JPEG2000 test image, Woman.

Tables 3, 4, 5 and 6 summarize the test results for the three commonly used images, the 1096 images in the CorelDraw database, the eight medical images, and the eight JPEG2000 test images, respectively. Note that the block size and the embedding level (defined in Section 1 and further discussed later in this section) for Tables 5 and 6 are the same as those used in Table 1 and 2. This was done in order to make the results compatible to that achieved by using the algorithm in [14]. For the same consideration, 100 information bits were embedded for the experiments shown in both Tables 1 and 5, because 100 information bits are also embedded in 512 by 512 grayscale images in [14] (majority voting is used, which will be explained later in this section).

Table 3. Test results for commonly used $512 \times 512 \times 8$ grayscale images.

	Lena	Baboon	Boat
PSNR (dB)	40.2	38.7	40.5
Capacity (bits)	792	585	560
Robustness (bpp)	0.8	1.6	1.0

Table 4. Test results for all of 1096 images in CorelDraw database.

Images (512×768)	PSNR of marked image (dB)			Data embedding capacity (bits)	Robustness (bpp)		
	Max	Min	Avg	714	Max	Min	Avg
	45.2	37.4	40.2		2.0	0.2	1.21

Table 5. Test results for eight medical images with block size 8, embedding level 6.

Images (512×512)	PSNR of marked image (dB)	Data embedding capacity (bits)	Robustness (bpp)
Mpic1	37.6	100	0.4
Mpic2	37.7	100	0.8
Mpic3	37.6	100	0.4
Mpic4	37.6	100	0.8
Mpic5	37.6	100	0.4
Mpic6	37.6	100	1.2
Mpic7	37.6	100	0.4
Mpic8	37.6	100	0.8

Table 6. Test results for eight JPEG2000 color test images with block size 20, embedding level 8.

Images (1536×1920)	PSNR of marked image (dB)	Data embedding capacity (bits)	Robustness (bpp)
N1A (Woman)	41.5	1410	0.8
N2A	41.3	1410	1.2
N3A	41.3	1410	0.8
N4A	41.4	1410	0.8
N5A	41.3	1410	0.8
N6A	41.2	805	0.4
N7A	41.2	1410	0.8
N8A	41.2	805	1.2

In order to compare the performance between the modulo-256 based algorithm [14] and the proposed algorithm in a more accurate way, a set of experiments has been conducted on the eight medical images. Comprehensive test results are shown in Table 7. In this set of experiments, the block size and embedding level were changed to observe the embedding capacity, the PSNR of the marked images versus the

original images, and the robustness against image compression. The block size represents the number of pixels along one side of a square block. The embedding level denotes the amount of grayscale value change within the block for a bit '1' or '0' to be embedded into the block. As mentioned in our introduction to the algorithm reported in [14], a bit '1' is embedded by rotating the mass vector of zone A by an angle counterclockwise and rotating the mass vector of zone B by the same angle clockwise. The rotation of the vector by an angle is equivalent to adding or subtracting an amount of grayscale values to each pixel in zones A or B, respectively. This amount of grayscale value change is denoted as the embedding level.

It is apparent from the table that the larger the embedding level, the lower the PSNR of marked image and the stronger the robustness of hidden data against image compression (and vice versa). In order to make comparison accurate, the algorithm in [14] was used to embed 100 information bits into each image. When the number of blocks (and hence the embedding capacity) is large, the same 100 information bits were repeatedly embedded. After hidden data extraction, majority voting was used to decode the hidden information bits.

It is obvious that, for a given block size, the embedding capacity for the two different methods is the same. The PSNR of a marked image versus its original image depends on embedding level and on block size. This is because as block size increases the variance of the α value decreases, allowing a smaller threshold value K and shifting quantity β to be used. In Table 7, the PSNR values of the eight marked images generated using each method are averaged and listed in Table 8 for each combination of block size and embedding level. The robustness in terms of minimum surviving bit-rate is listed in Table 7. For each combination of block size and embedding level, the bit rates for the eight medical images are averaged and listed in Table 8. Furthermore, the PSNR and the minimum surviving bit rate are also averaged over different embedding levels for a given block size (marked by *), as well as over different block sizes (marked by **). This is listed in Table 8. It is observed from Table 8 that for each combination of block size and embedding level (and therefore each specified data embedding capacity)

the average PSNR over eight marked medical images using the proposed method is much higher than that obtained using the algorithm in [14]. It is also observed that for each combination of block size and embedding level, the average minimum surviving bit rate associated with the proposed method is lower than that with the algorithm in [14]. In other words, the average robustness against image compression with the proposed method is stronger than that with the algorithm in [14]. In summary, better performance has been achieved with the proposed method, compared with that in [14].

Table 7. Test results for various combinations of block size and embedding level by using the proposed method and the method in [14] on eight medical images. (The listed robustness against image compression is the minimum surviving bit rate in terms of bpp. Refer to text.)

			Mpic1		Mpic2		Mpic3		Mpic4		Mpic5		Mpic6		Mpic7		Mpic8	
	Block size	Embed level	PSNR (dB)	Robustness	PSNR (dB)	Robustness	PSNR (dB)	Robustness	PSNR (dB)	Robustness	PSNR (dB)	Robustness	PSNR (dB)	Robustness	PSNR (dB)	Robustness	PSNR (dB)	Robustness
Proposed algorithm	8	4	40.5	0.4	40.7	0.8	40.6	0.8	40.6	0.8	40.6	0.8	40.5	1.2	40.5	0.4	40.5	0.8
		6	37.6	0.4	37.7	0.8	37.6	0.4	37.6	0.8	37.6	0.4	37.6	1.2	37.6	0.4	37.6	0.8
		8	35.4	0.4	35.6	0.8	35.5	0.4	35.4	0.4	35.4	0.4	35.5	0.8	35.4	0.4	35.4	0.4
		10	33.7	0.4	33.8	0.4	33.7	0.4	33.7	0.4	33.7	0.4	33.7	0.8	33.7	0.4	33.7	0.4
	12	4	40.7	0.4	40.9	0.8	40.7	0.4	40.7	0.8	40.7	0.8	40.8	1.2	40.7	0.4	40.7	0.8
		6	37.8	0.4	38	0.8	37.8	0.4	37.8	0.8	37.8	0.4	37.9	1.2	37.8	0.4	37.8	0.8
		8	35.5	0.4	35.6	0.8	35.7	0.4	35.6	0.4	35.6	0.4	35.7	0.8	35.6	0.4	35.6	0.4
		10	33.8	0.2	34.1	0.4	33.8	0.2	33.8	0.4	33.8	0.2	34	0.8	33.8	0.2	33.8	0.4
	16	4	40.6	0.4	40.9	0.8	40.6	0.4	40.6	0.8	40.6	0.4	40.7	1.2	40.6	0.4	40.5	0.8
		6	37.7	0.2	38	0.4	37.7	0.2	37.7	0.4	37.7	0.2	37.8	0.8	37.6	0.2	37.6	0.4
		8	35.5	0.4	35.9	0.8	35.5	0.4	35.5	0.4	35.5	0.4	35.6	0.8	35.5	0.4	35.4	0.4
		10	33.8	0.2	34	0.4	33.8	0.2	33.8	0.4	33.8	0.2	33.9	0.8	33.8	0.2	33.7	0.4
Algorithm in [9]	8	4	9.9	1.6	4.9	1.6	29.0	1.2	29.1	0.8	29.1	1.6	5.8	2.0	10.1	0.8	6.2	1.2
		6	9.9	0.8	4.9	1.6	28.1	0.8	28.2	0.4	28.2	0.8	5.9	2.0	10.5	0.8	6.3	1.6
		8	9.9	0.8	5.0	2.0	27.1	0.8	27.2	0.4	27.2	1.6	5.9	1.6	10.3	0.8	6.3	2.0
		10	9.8	1.6	5.1	1.6	26.1	0.8	26.2	0.4	26.2	0.8	6.0	2.0	7.8	0.8	6.4	2.0
	12	4	10.1	0.8	5.0	1.6	28.0	0.8	28.2	0.8	28.6	1.2	6.0	2.0	10.8	0.8	6.4	1.6
		6	10.1	0.8	5.1	1.6	27.2	0.4	27.5	0.2	27.8	0.2	6.0	2.0	10.8	0.4	6.4	1.6
		8	10.1	0.8	5.1	2.0	26.4	0.8	26.6	1.6	26.8	0.4	6.1	2.0	10.6	0.8	6.5	0.8
		10	9.9	0.8	5.2	2.0	25.6	2.0	25.7	0.4	26.0	0.2	6.1	2.0	8.0	0.8	6.5	0.8
	16	4	10.6	0.8	5.3	1.6	29.0	0.8	29.2	0.8	29.2	0.4	6.4	1.2	11.2	0.8	6.9	1.2
		6	10.6	0.8	5.4	0.8	28.2	2.0	28.3	0.2	28.3	0.2	6.4	0.8	11.3	0.4	7.0	0.8
		8	10.6	0.8	5.4	1.6	27.3	0.8	27.4	0.4	27.4	0.4	6.5	1.6	11.1	0.8	7.0	0.8
		10	10.4	0.8	5.5	0.8	26.3	0.2	26.4	0.2	26.4	0.2	6.5	0.8	8.1	0.4	7.0	0.8

Table 8. Performance comparison on PSNR of marked image versus original image and robustness against image compression averaged over the eight medical images between the proposed method and the method in [14]. (For the meaning of * and **, refer to text.)

	Block size	Embedding level	Average PSNR (dB)			Average robustness (bpp)		
Proposed algorithm	8	4	40.5	36.8*	36.9**	0.75	0.59*	0.53**
		6	37.6			0.65		
		8	35.4			0.5		
		10	33.7			0.45		
	12	4	40.7	37.0*		0.7	0.55*	
		6	37.8			0.65		
		8	35.6			0.5		
		10	33.8			0.35		
	16	4	40.6	36.9*		0.65	0.46*	
		6	37.7			0.35		
		8	35.5			0.5		
		10	33.8			0.35		
Algorithm in [9]	8	4	15.5	15.0*	15.1**	1.35	1.24*	1.04**
		6	15.2			1.1		
		8	14.8			1.25		
		10	14.3			1.25		
	12	4	15.4	14.8*		1.2	1.09*	
		6	15.0			0.9		
		8	14.7			1.15		
		10	14.0			1.125		
	16	4	16.0	15.4*		0.95	0.78*	
		6	15.7			0.75		
		8	15.3			0.9		
		10	14.6			0.525		

2.9 Conclusion and Discussion

A novel robust lossless image data hiding scheme has been proposed that employs a robust statistical quantity to mitigate the effect of image compression and small incidental alteration. It utilizes different bit-embedding strategies for groups of pixels with different pixel grayscale value distributions. It employs error correction codes together with a permutation scheme to minimize errors. Consequently, it has successfully avoided using modulo-256 addition to achieve losslessness, thus eliminating annoying salt-and-pepper noise.

This technique has the following advantages: 1) no salt-and-pepper noise; 2) applicable to virtually all images (the algorithm has been successfully tested in some commonly used images, eight medical images, all of 1096 images in the CorelDRAW database, and all of eight JPEG2000 test images); 3) average PSNR of marked images is above 38 dB; 4) robust to JPEG/JPEG2000 compression; 5) data embedding capacity ranges from 512 bits to 1024 bits (often sufficient for authentication purpose), and, 6) the embedding capacity can be adjusted according to the requirement.

This proposed scheme [19] has been utilized to embed digital signature related data to authenticate losslessly compressed JPEG2000 images, followed by possible transcoding [20]. A unified authentication framework, proposed and described in [20, 21] provides both fragile and semi-fragile authentication. The former, fragile authentication, is used for data integrity verification while the latter, semi-fragile authentication, is used for content integrity verification. Within semi-fragile authentication there are two different modules, lossy and lossless. The robust lossless data hiding scheme reported here is used for the lossless module. If a losslessly compressed JPEG2000 image has not been altered before authentication, the hidden data can be extracted correctly, the image will be classified as authentic and the original image can be recovered exactly. If the losslessly compressed JPEG2000 image has experienced further transcoding (for instance, lossy compression), it will be rendered authentic as long as the compression is not so severe that the content has been changed. At this point, the hidden data can be extracted correctly, but the original image will not be able to be recovered. If the lossy compression is so severe that the resultant bit rate is lower than the specified minimum surviving bit rate, the hidden data will not be extracted correctly and the image will be rendered non-authentic. If the content of the losslessly compressed image has been altered and the hidden data can be extracted without error, the extracted data will render the altered image non-authentic

because the hidden data does not match the altered content. For detail, please refer to [20]. This unified authentication framework for JPEG2000 images has been included in the JPEG Security Part (known as JPSEC) FCD (Final Committee Draft) version 1.0 in November 2004 [21].

3. A Novel Robust Lossless Digital Watermarking Scheme Based on Integer Wavelet Transform

As stated in Section 1, De Vleeschouwer et al. proposed a semi-fragile lossless data hiding scheme using modulo-256 addition [14]. It divides the cover image into non-overlapping blocks, with pixels in each block randomly grouped into two sets of equal size. The pixel grayscale values of each set are projected onto a circle and the mass center of the circle is calculated, as in the spatial technique described in Section 2. High correlation between pixels in the block results in similar the positions of the mass centers for these two sets with respect to the corresponding circles. For each mass center, a vector is formed from the circle center to the mass center. By manipulating the orientation of these two vectors, one bit can be embedded into the block. As the mass center position is stable against slight alterations, this embedding scheme can survive the JPEG compression to a certain extent.

However, one big problem with this method is that modulo-256 addition is used to prevent overflow/underflow in order to achieve losslessness. Consequently, pixels with their grayscale values close to 255 may be flipped after modulo-256 addition to pixels with grayscale values close to zero and vice versa, resulting in salt-and-pepper noise. The salt-and-pepper noise can be quite annoying for some images, such as those with high contrast. Two examples have been shown in Figures 3 and 4.

Recently, JPEG2000 has emerged as the newest standard for still image compression because of its superior efficiency. It supports both lossy and lossless compressions. In lossless compression, the 5/3 wavelet filter is used [22]. In this section, we propose a new semi-fragile lossless data hiding scheme based on the 5/3 integer wavelet transform (IWT). Data is embedded into the IWT coefficients of a selected high frequency subband. The subband accommodating the hidden data is called the *carrier*

subband. One advantage of the proposed IWT-based algorithm is that it can be integrated into JPEG2000 standard smoothly. The hidden data can be extracted from the JPEG2000 files without transforming the image file to the spatial domain, which reduces the computational burden. Benefits of this method include the survival of hidden data after incidental alterations, and the recovery of the original cover image if the stego-image is not subject to any alterations. The main advantage of the proposed scheme is that no annoying salt-and-pepper noise will appear in the stego-images, and the visual quality is hence rather high.

3.1 Foundation

The features of image wavelet transforms have been studied and it has been found that coefficients of the high frequency subbands (HL_n, LH_n, or HH_n) have a zero-mean and Laplacian-like distribution. One such example, the HL_1 subband of the N1A JPEG2000 test image, is depicted in Figure.24. Here the X-axis represents the IWT coefficient values, while the Y-axis the numbers of coefficients that assume the corresponding value. One can further divide this subband into non-overlapping square blocks with side length B and calculate the mean of the coefficients values in each block.

$$Mean^{(t)} = \frac{1}{B^2}\sum_{i=1}^{B}\sum_{j=1}^{B} C_{ij}^{(t)} \qquad (1)$$

where the superscript $^{(t)}$ indicates the t-th block. $C_{ij}^{(t)}$ denotes the IWT coefficients of the t-th block.

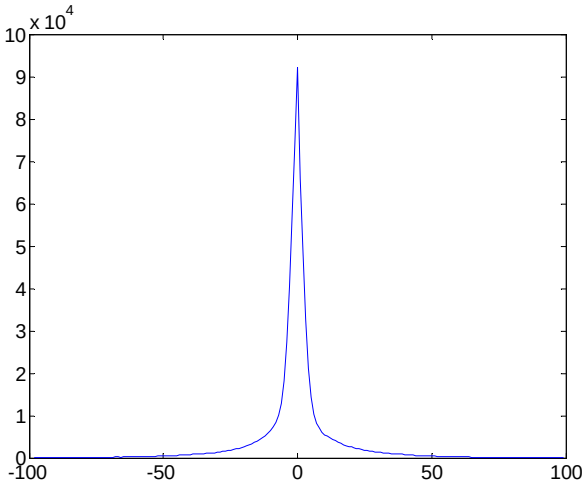


Figure. 24 Histogram of the IWT coefficients in the HL_1 subband of N1A (red color plane). The size of N1A is 1536×1920. For the HL_1 subband, the size is 768×960, i.e., having 737,280 coefficients.

Figure.25 illustrates one example of the distribution of these mean values. There, X-axis is the mean values of 10×10 non-overlapping blocks in the HL_1 subband of the N1A JPEG2000 test image, and Y-axis is the number of blocks that have a particular mean value. It can be observed that the variance of the mean values shown in Figure.7 is much smaller than that of the coefficients shown in Figure.25.

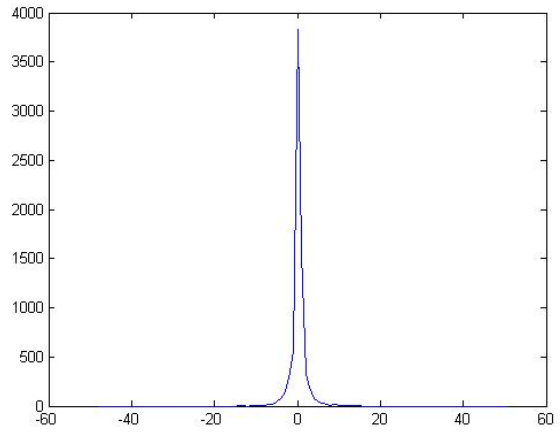


Figure.25 Mean value distribution of 10x10 blocks in the HL_1 subband of N1A (red color plane).

The basic idea of the proposed semi-fragile lossless data hiding algorithm is to exploit this property. One bit of data will be embedded into each of the non-overlapping blocks, performed in the following manner. First, all the blocks are scanned and the maximum absolute mean value of coefficients is found, denoted by m_{max}.

$$m_{max} = \underset{t}{Max}(|\, mean^{(t)} \,|) \tag{2}$$

The variable *t* denotes the block index, and *Max* the operation to take maximum value. A threshold *T* is set to be the smallest integer number which is no less than m_{max}, resulting in the absolute mean values of all blocks being smaller than *T*. One bit can then be embedded into each of the non-overlapping blocks by manipulating the mean value of the block. If a bit '1' is to be embedded, the mean value of the block will be shifted away from *0* by a quantity *S*, which is equal or larger than *T*. If a bit '0' is to be embedded, this block is left unchanged, as is the mean value of the IWT coefficients in that block. Therefore, the mean value of a block with an absolute value larger than *T* represents a hidden binary '1', and conversely a mean value smaller than *T* represents an embedded binary '0'.

Since the shift quantity *S* is fixed for all blocks, the original coefficients can be recovered by conducting the reverse operation, i.e., subtracting *S* from the IWT coefficients in the blocks where bit '1' is embedded. The embedding operation is thus reversible, and as a result the cover media can be restored without any error if no distortion has taken place to the stego-image. Since data is embedded by controlling the mean value of the IWT coefficients in one block, minor changes to the image caused by unintentional attacks (such as JPEG/JPEG2000 compression) will not cause the mean value to change much, allowing for the correct detection of the hidden data even after the stego-image has been altered slightly. This claim will be verified in Section V by presenting experiment results.

It should be pointed out that this lossless watermarking algorithm is based on a principle of information theory which relies on all information in the original cover media to be preserved in the marked image after the lossless data hiding process. Therefore, if the marked image is subjected to lossy compression

the original cover media can no longer be recovered since information about the original image will be impaired. It should also be noted that only bands HL_1 or LH_1 were chosen to embed data. Although the mean value distribution of blocks of IWT coefficients for higher level high frequency subbands (HL_n or $LH_n(n>1)$) also obey the observation described in this section, alterations to those higher subbands will degrade the image quality dramatically.

3.2 Proposed Lossless Data Hiding Algorithm in Integer Wavelet Domain

In this section the issue of preventing overflow/underflow is addressed, which is a key issue in lossless data hiding. The proposed data embedding procedure is then presented.

A. Origin of overflow/underflow

For images in the JPEG2000 format, the proposed method in the previous section is good enough, For example, an image in the JPEG2000 format with all the IWT coefficients derived from the image file through tier 2 decoding followed by tier 1 decoding [23] could be used for embedding by placing the hidden data into the IWT coefficients as described above. The changed coefficients can be encoded again to form the marked JPEG2000 file and used in applications such as transmission. At the receiver end, the hidden data could be extracted from the marked JPEG2000 image file. If no attack has occurred to the marked image than all the modified coefficients can be inverted back to the original ones with the knowledge of how much the coefficients have been shifted away from the origin in the data embedding, i.e., the above-mentioned parameter S. The recovered version of the image will be exactly the same as the original one. This is because the JPEG2000 encoding ensures no truncation to the wavelet coefficients [24], and therefore overflow and/or underflow will not take place.

However, sometimes the stego-image may be converted into formats other than the JPEG2000 format. File format change should be considered permissible, and the original cover media should be preserved after the file format change. In doing so, the recovered media may differ from the original after format changes, such as conversion from the JPEG2000 format to the bitmap format. This is because the pixel

grayscale values in spatial domain are represented by a fixed length of bits (normally 8 bits, sometimes 16 bits). In color images, each fundamental color component in a pixel is also often represented by 8 bits. After the IWT coefficients have been altered by the embedding operation, some pixels' grayscale values will fall out of the permitted range, (0~255 for an 8-bit grayscale image). This phenomenon is referred to as overflow/underflow. In other words, since some IWT coefficients are modified in the embedding process using this algorithm, if the JPEG2000 file is transformed to the spatial domain then the pixel grayscale values may exceed the range from 0 to 255. As a result, truncation will have to be implemented. Pixel grayscale values that are greater than 255 (overflow) will be forced to equal to 255 and those smaller than zero (underflow) will be forced to 0. From an information theory perspective there is information loss in truncation, and therefore the original image cannot be recovered correctly later after the data extraction.

B. Prevention of overflow/underflow (I)

From the above discussion, it has been shown that as long as the pixel grayscale values of the marked image in the spatial domain remain in the permitted range, the overflow/underflow problem can be avoided. In the rest of this paper, it will be assumed that the pixel grayscale value is represented by 8 bits, for a permitted range of [0,255]. If the permitted range is not [0,255] for some images, the proposed principle is still valid and the corresponding solution can be easily derived.

As previously stated, the proposed scheme is block based; specifically, it splits the selected high frequency subband into non-overlapping blocks and the data hiding is carried out block by block. However, the modification of the IWT coefficients in the blocks adjacent to a block under consideration will interfere with the grayscale values of the pixels corresponding to this block. This phenomenon will be further discussed in Part C of this section. The point made here is that it is much easier to avoid overflow/underflow if this type of interference among neighboring blocks is removed. Therefore, the following mask is introduced.

$$\begin{pmatrix} 0 & 0 & 0 & \cdots & 0 & 0 & 0 \\ 0 & 1 & 1 & \cdots & 1 & 1 & 0 \\ 0 & 1 & 1 & \cdots & 1 & 1 & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ 0 & 1 & 1 & \cdots & 1 & 1 & 0 \\ 0 & 1 & 1 & \cdots & 1 & 1 & 0 \\ 0 & 0 & 0 & \cdots & 0 & 0 & 0 \end{pmatrix} \quad (3)$$

The outer elements on the bound of the mask are marked by '0's and the inner elements by '1's. In the mask, the '0' denotes an IWT coefficient, which will not be changed, while the '1' a coefficient which will be changed in data embedding. In other words, in order to change the mean value of an IWT-coefficient block during bit embedding only those coefficients whose positions are associated with '1' in the above mask will be allowed to change, keeping the coefficients intact on the outer bound of the block. With IWT theory [23], it is easy to prove that when the IWT coefficients are changed in one block using the manner characterized by the mask shown in Formula (3), the pixel grayscale values in adjacent blocks will not be affected. That is, the interference among neighboring blocks will be eliminated.

After the interference among adjacent blocks has been removed, focus can be placed on one single block in preventing overflow/underflow. The basic idea is described as follows: Consider an IWT-coefficient block, B_w, and its corresponding block in the spatial domain, B_s (Here "corresponding" means the group of pixels whose grayscale values are associated with the IWT-coefficient block). According to IWT theory [22,25], this correspondence relation can easily be determined. The maximum absolute pixel grayscale value change is denoted S_{max}. In B_s, overflow may occur if there are pixels with their grayscale values greater than $(255 - S_{max})$ and the grayscale values need to be increased in the bit embedding process. Underflow may take place when there are pixels with their grayscale values less than S_{max} and the grayscale values need to be decreased. To avoid overflow and underflow, the above two scenarios should not occur. We call $[0, S_{max}]$ as *0-zone* and $(255\text{-}S_{max}, 255)$ as *255-zone*. A block can be classified into one of the following four categories according to the presence and/or absence of these two different zones in the spatial domain.

The best case is one where none of the above two zones are present, as is illustrated in Figure 26(a). In this case, we do not need to worry about the overflow/underflow problem. A second case occurs if in B_s there exist pixels in *0-zone* but none in *255-zone*, as is illustrated by the solid line in Figure 26(b). It is deemed "safe" if all changes to the pixel grayscale values in this group are restricted to increase only. The resultant histogram after data hiding is represented by the dotted line in Figure 26(b). Under these circumstances, it is evident that no pixel will have the grayscale value greater than 255 or smaller than zero, indicating no overflow/underflow will occur. Figure 26(c) depicts the opposite case to the previous one. If there is no pixel of this block in the *0-zone*, it will be safe if all changes to the pixels in this group are decreases. The worst case is described by Figure 26(d) where pixels in *0-zone* and pixels in *255-zone* are both presented in one block. In this case, it is noted that the coefficients of this block will not be modified in data embedding in order to avoid overflow and underflow. Thus, errors in data extraction may take place and some additional measure must be taken. This will be discussed later in this report.

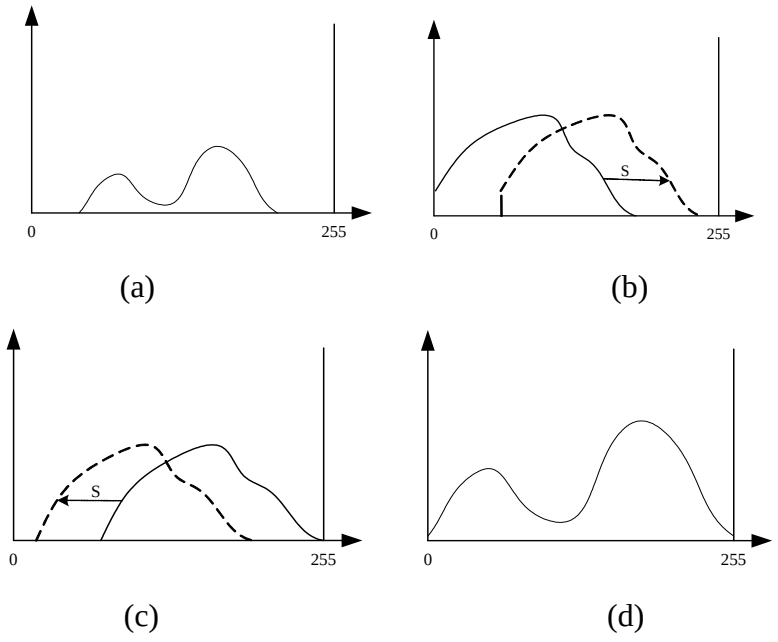


Figure 26. Four categories of blocks.

C. Prevention of overflow/underflow (II)

In the preceding sections a mechanism has been proposed to embed data bits into IWT coefficients and the avoidance of overflow/underflow by considering the tendency of grayscale change has been discussed. If a way to embed one bit into one block of the HL_1 or LH_1 subband can be found such that all of the affected pixels will only move their grayscale values toward the desired direction, the overflow/underflow problem can be avoided. In this subsection, this issue is discussed further.

As previously mentioned, in the JPEG2000 standard the 5/3 wavelet filter is used as the default filter for reversible image encoding [22,23]. The 5/3 filter was first proposed by Le Gall et. al in [26]. The coefficients of the 5/3 wavelet filter are given in Table 9 [22].

Table 9. 5/3 filter coefficients.

	Analysis Filter Coefficients		Synthesis Filter Coefficients	
i	Low-pass Filter $h_L(i)$	High-pass Filter $h_H(i)$	Low-pass Filter $g_L(i)$	High-pass Filter $g_H(i)$
0	6/8	1	1	6/8
±1	2/8	-1/2	1/2	-2/8
±2	-1/8	0	0	-1/8

The goal of this research is to embed data into a high frequency subband of the integer wavelet transform (IWT) of an image. To do so, the effects of changing the wavelet coefficients on the spatial domain counterpart need to be investigated. Table 9 shows the unit step response in the spatial domain for a change made to an IWT coefficient located at the (i, j) position in the HL_1 subband. This can be expressed with the matrix G_{HL} in Formula (4). Specifically, an array of pixel grayscale values in the spatial domain centered at (2i-1, 2j) with a size of 3×5 (refer to Figure 27) will change by the amount specified by the elements of this matrix if the IWT coefficient (i, j) in the HL_1 subband is increased by one.

$$G_{HL} = \begin{bmatrix} -\frac{1}{16} & -\frac{1}{8} & \frac{3}{8} & -\frac{1}{8} & -\frac{1}{16} \\ -\frac{1}{8} & -\frac{2}{8} & \frac{6}{8} & -\frac{2}{8} & -\frac{1}{8} \\ -\frac{1}{16} & -\frac{1}{8} & \frac{3}{8} & -\frac{1}{8} & -\frac{1}{16} \end{bmatrix} \quad (4)$$

The corresponding unit step responses can be found for the coefficients in LH_1 and LL_1 subbands, denoted by G_{LH} and G_{LL} respectively, as shown below. They indicate the amount of change to the grayscale values in special domain if the IWT coefficient (i, j) in the LH_1 or LL_1 subband is increased by one. The center is located at (2i, 2j-1) and (2i-1, 2j-1) for G_{LH} and G_{LL}, respectively.

$$G_{LH} = \begin{bmatrix} -\frac{1}{16} & -\frac{1}{8} & -\frac{1}{16} \\ -\frac{1}{8} & -\frac{2}{8} & -\frac{1}{8} \\ \frac{3}{8} & \frac{6}{8} & \frac{3}{8} \\ -\frac{1}{8} & -\frac{2}{8} & -\frac{1}{8} \\ -\frac{1}{16} & -\frac{1}{8} & -\frac{1}{16} \end{bmatrix} \quad (5) \qquad G_{LL} = \begin{bmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{2} & 1 & \frac{1}{2} \\ \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \end{bmatrix} \quad (6)$$

Assuming that HL_1 is the carrier subband, Formula (4) indicates that the pixels affected by wavelet coefficient change will change their grayscale values in both increasing and decreasing directions. Similarly, this is true if LH_1 is used as the carrier subband. According to Formula (6), the pixel grayscale value changes resulting from wavelet coefficient change will move toward only one direction. The core idea of combating overflow/underflow is to manipulate coefficients in the LL subband to force the resultant pixel grayscale value change toward only one desired direction.

By studying the unit response of LL coefficients it can be deduced that for an LL_1 coefficient at (i, j) the center of the unit step response is at (2i-1, 2j-1) in the spatial domain, while the center of unit step response will be at (2i-1, 2j+1) for LL_1 coefficient (i, j+1). Figure 27 illustrates the relationship of an IWT coefficient and its unit step response in the spatial domain. The solid arrow and the solid box are for the

coefficient (i, j) in HL_1 subband; the dotted arrow and box are for LL_1 coefficient (i, j); the dashed arrow and box are for LL_1 coefficient (i, j+1).

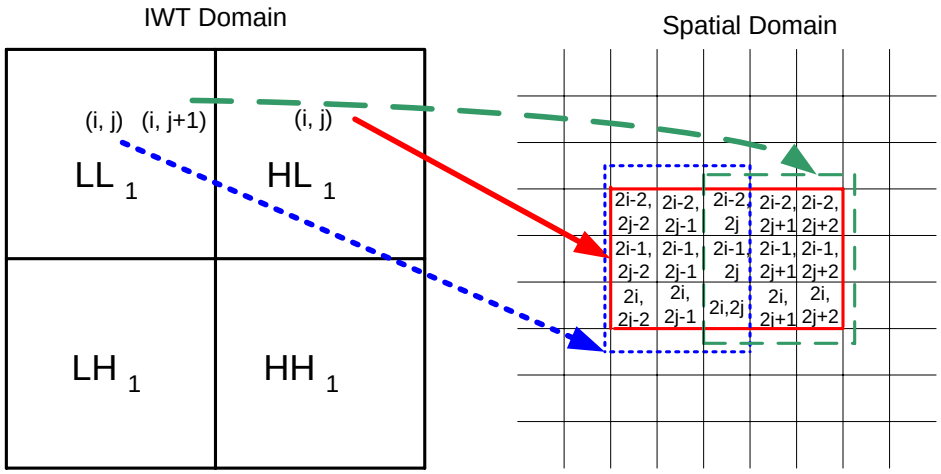


Figure 27. Relationship between IWT coefficients and pixel in spatial domain

It is observed that the combined 2-D arrays corresponding to the unit step responses of LL_1 coefficients (i, j) and (i, j+1) have the size of 3×5 and cover the exactly same group of pixels corresponding to the unit step response of the IWT HL_1 coefficient (i, j). $C_{HL}(i, j)$ will be used to denote the coefficient at (i, j) of the HL_1 subband, and $C_{LL}(i, j)$ the coefficient at (i, j) of the LL_1 subband. It is straightforward to verify that the combined effect of the unit step response of $C_{HL}(i, j)$, the 1/4 of the unit step response of $C_{LL}(i, j)$ and the 1/4 of the unit step response of $C_{LL}(i, j+1)$ can be represented by Formula (7).

$$R = \begin{bmatrix} 0 & 0 & \frac{1}{2} & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 & 0 \end{bmatrix} \qquad (7)$$

Formula (7) indicates that all the change in the spatial domain will have the same sign. In other words, if data is to be embedded in the HL_1 subband, an HL_1 coefficient at (i,j) is changed by S and both LL_1

coefficients at (i,j) and (i,j+1) are changed by S/4. The pixel grayscale value change in the spatial domain will then have the same sign as that of S. It is noted that after the embedding, the maximum pixel grayscale value change in the spatial domain is also S. Similarly, if the LH_1 serves as the carrier subband, the same goal can be achieved by changing the LH_1 coefficient at (i,j) by S and both the LL_1 coefficients at (i,j) and (i+1,j) by S/4. The next section presents the proposed data embedding procedure and clarifies how overflow and underflow are avoided by utilizing features presented in Parts B and C.

D. Semi-fragile Lossless Watermarking Algorithm

This is a summarization of the theoretical results derived in the previous section. Suppose the HL_1 subband is used as the embedding carrier. If the coefficient at (i, j) in the HL_1 subband is changed by a quantity S (S can be either positive or negative), modify two coefficients of LL_1 subbands at (i, j) and (i, j+1) each by a quantity S/4. As a result, all the affected pixels in the corresponding 3×5 array will either increase (if S>0) or decrease (if S<0). Alternatively, if the LH_1 subband is used as the embedding carrier and the coefficient at (i, j) in LH_1 subband is modified by S, the two corresponding coefficients of LL_1 subbands at (i, j) and (i+1, j) will need to be changed by S/4. Similarly, all the affected pixels in the spatial domain will either increase (if S>0) or decrease (is S<0). As mentioned previously, in this algorithm the mean value of a block is shifted by changing all the coefficients with mask value '1' in Formula (3) by the same amount. For each coefficient in the HL_1 or LH_1 subband whose value needs to be changed by S, in addition to changing its value by S it is also necessary to change the two corresponding coefficients in the LL_1 subband by S/4 as discussed above.

Before embedding, error correction coding (ECC) [16] and chaotic mixing [18] (described in Section 2) is applied to the information bits to be embedded. The purpose of applying these two techniques is to improve the robustness of this proposed algorithm. The coded bit stream and the cover image are the input to the embedding algorithm.

When embedding a bit into a block of IWT coefficients in the HL_1 or LH_1 subband, denoted by B_w, it is necessary to first find the corresponding spatial block that could possibly be affected by the embedding, denoted B_s. The pixel grayscale values of B_s are then checked and the block is classified into one four categories, which have been depicted in Figure 26.

A: Pixels with no grayscale values greater than ($255-S_{max}$) or smaller than S_{max}

B: Pixels with grayscale values smaller than S_{max}, but no pixel values greater than ($255-S_{max}$)

C: Pixels with grayscale values greater than ($255-S_{max}$), but no pixel values smaller than S_{max}

D: Pixels with both grayscale values smaller than S_{max} and greater than ($255-S_{max}$)

Different operations are applied to different categories. For each category, the corresponding operation is presented below.

1) Blocks of type A

In this case, overflow and underflow will not take place. The mean value of the carrier subband (HL_1 or LH_1) coefficients in the block can either be increased or decreased. The mean value may be greater than zero or less than zero, as is depicted in Figure 26(a). This is illustrated in detail in Figure 28. If ‘1’ is to be embedded, the mean value will be shifted away from 0 by S, i.e., from a or b according to the sign to a’ or b’ of the original mean. The corresponding coefficients in the LL subband can be modified by S/4 as described in Part C of this section. For bit ’0’, the mean of this block will remain unchanged.

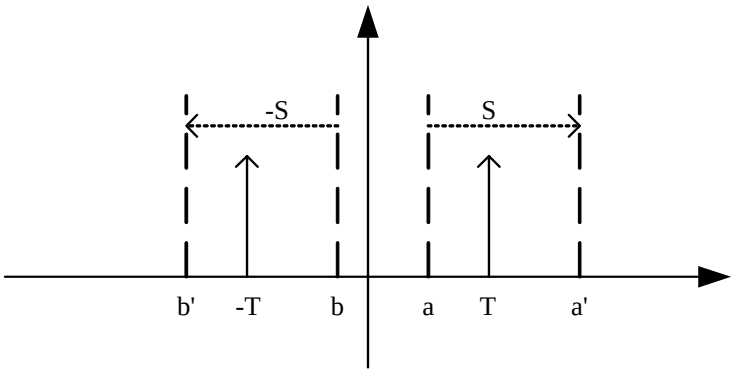


Figure 28. Embedding rule for type A block.

2) For blocks of type B

Blocks of this type have some pixel grayscale values in the *0-zone* as is depicted in Figure 26(b). Overflow will not occur for this type of block. To embed a bit ‘0’, nothing needs to be done. To embed a bit ‘1’, the absolute value of the mean should be shifted by S. If the original mean value is greater or equal to zero, as is the case of a in Figure 29, the mean can be shifted to a’. The corresponding coefficients in the LL subband will be modified by S/4. As the result, the pixel values in the spatial domain will only increase after changes are made, therefore avoiding underflow. If the mean value is, for example, negative b to make the absolute mean value greater than T, the mean value can only be subtracted by T. However, underflow may happen. Therefore, the mean value will not be shifted (which is equivalent to embedding a bit ‘0’). Therefore, error correction coding is introduced, i.e., ECC decoding is relied upon to correctly recover the hidden bits.

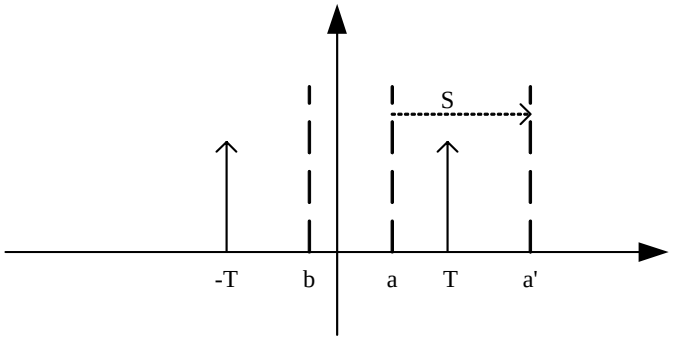


Figure 29. Embedding rule for type B block.

3) For block of Type C

The blocks of this type have some pixel grayscale values in the *255-zone* as is depicted in Figure 26(c). This case is similar to the previous one (Type B) except everything is the opposite. To embed bit ‘0’, no operation is needed. For bit ‘1’, only the blocks with a negative mean can be used to hide bit ‘1’. If bit ’1’ is to be embedded and the mean value of the block is positive, it is not possible to embed the bit. Instead, this block is not operated on, which is equivalent to embedding bit ‘0’. An error bit will occur, requiring the ECC to correct errors. Figure 30 illustrates this in detail.

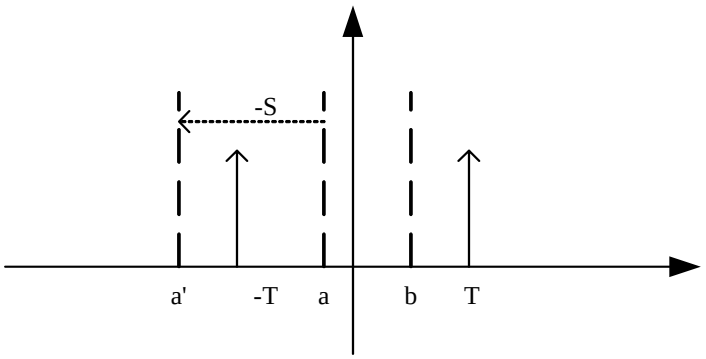


Figure 30. Embedding rule for type C block.

4) For block of Type D

Blocks of type D have some pixels whose grayscale values are in *0-zone* and some grayscale values in *255-zone* simultaneously, as is illustrated in Figure 26(d). Changes on the mean value in either direction will cause either overflow or underflow. If an information bit '0' is to be embedded here, there will be no problem. If it is required that a bit value '1' is embedded, an error will occur. Again, ECC decoding is relied upon to rectify the error.

It is noted that for frequently used images most blocks belong to type A. For example, all blocks in the "Lena" image are of Type A. For other types of blocks, ECC is necessary to be used to correct the errors. In some cases, the errors may cluster together since neighboring pixels have similar properties. This phenomenon is called a burst error in the literature and occurs when blocks of type B, C, D concentrate in some region(s) of a given image. In this circumstance, ECC cannot efficiently correct these errors [17]. Chaotic mixing [18] is introduced to combat burst errors, as it can disperse the burst errors evenly. Figure 31 provides the block diagram of the data embedding.

Block size B, threshold T, shifting value S and which subband is chosen as the carrier are saved as side information which will be needed for data extraction and cover media recovery later. From a different perspective, the side information can be treated as a key and transmitted through a separate secure channel. Only an authorized person can extract the hidden data with the provided side information. An alternative way of applying the proposed algorithm is to use the same B, T, S and subband for all images. As a result the side information would no longer be needed. The drawback of doing so is that the parameters used for embedding will not be the optimal for a given particular image.

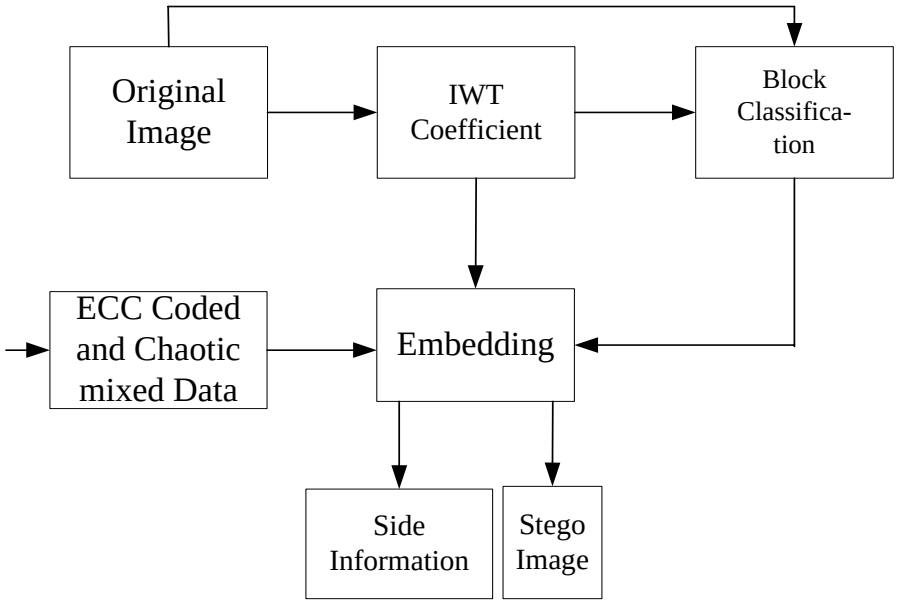


Figure 31. Block diagram of data embedding.

3.3 Data Extraction and Original Image Recovering

At the receiver side, a stego-image is the input. Data extraction and original cover image restoration will be performed.

A. Hidden data extraction

Hidden data extraction requires the stego-image and the side information. First, the integer wavelet coefficients of the stego-image are obtained. The carrier subband is then divided into non-overlapping blocks with size B in the same fashion as was done for embedding. From each block, one bit of data is extracted. At this stage, it is not important as to which category a block falls into. The comparison of the mean value with the threshold T will decide whether bit ‘0’ or bit ‘1’ is extracted. It is noted that if the test image is given in JPEG2000 format, the IWT coefficient can be obtained without performing an inverse wavelet transform. The data extraction will be efficient since no extra transformations are needed.

B. Cover media restoration

After all hidden data have been extracted, the original cover media can be recovered. Two possibilities exist. One is the case where stego-image has not subjected to any alteration. In this case, the original

image can be perfectly recovered. Figure 32 is an illustrative flowchart for recovering the original image. The coefficients of the original image are restored by shifting the coefficients' values of the stego-image towards the opposition direction to that in the embedding stage.

The other is the case where some alteration has been applied to the stego-image. It is noted that in this case the original cover image cannot be recovered because some information of the original image has been lost due to the alteration.

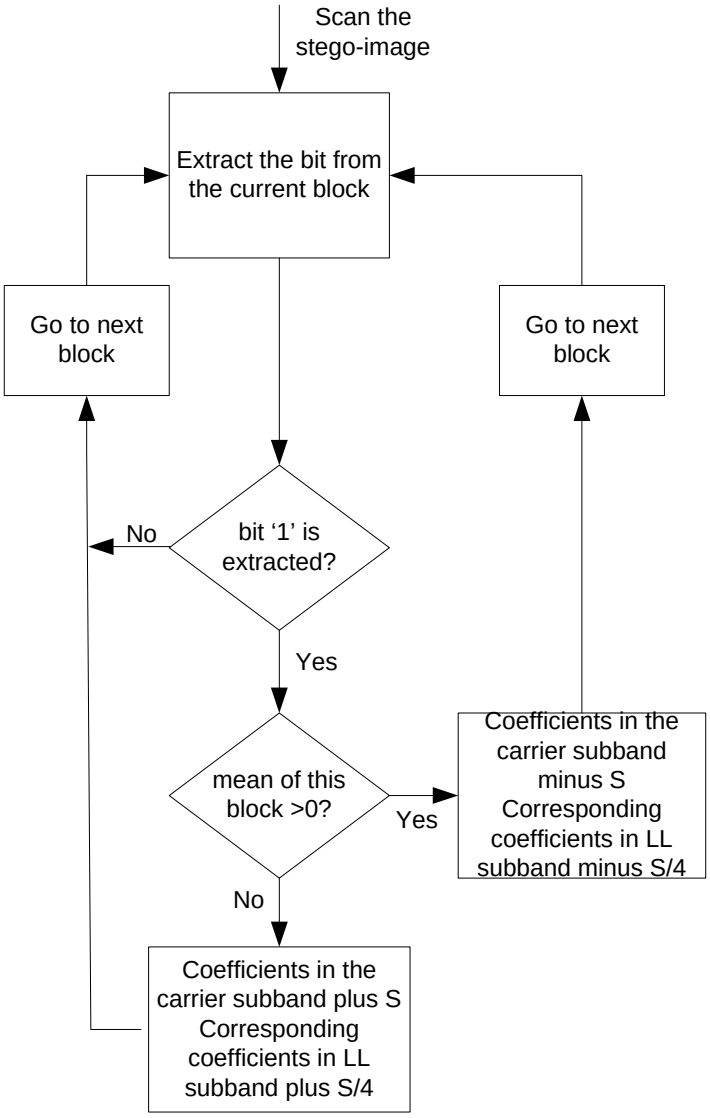
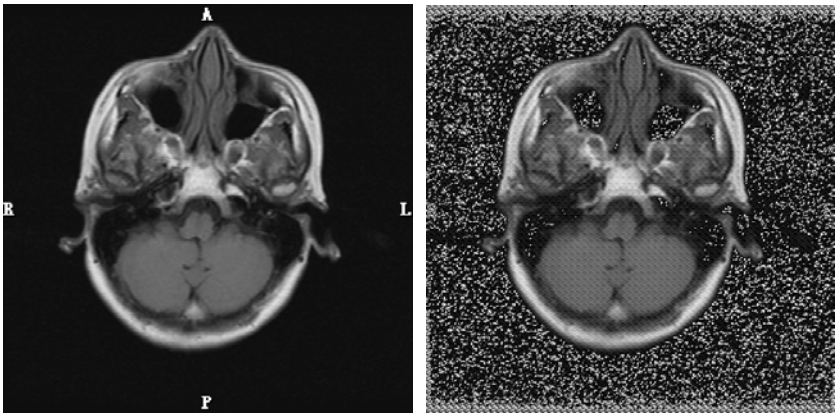


Figure 32 Cover media restoration flowchart.

3.4 Experimental Results and Performance Analysis

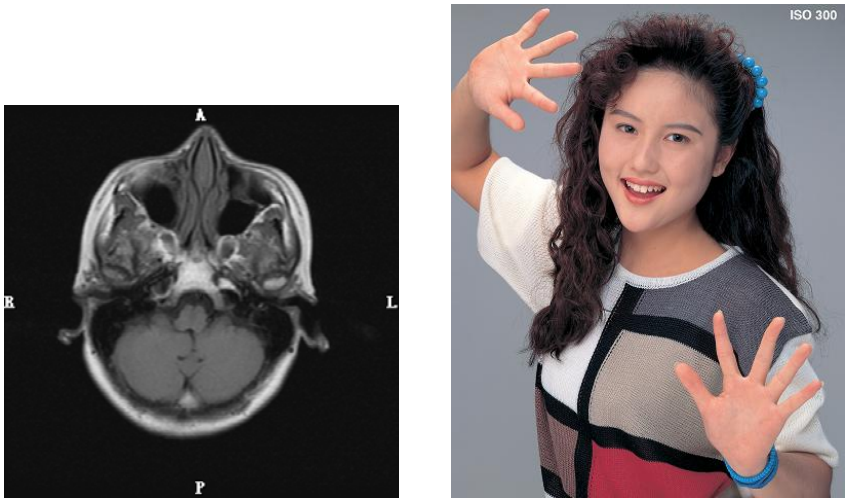
In the following experiments, for the sake of generality the bitstream to be embedded has been randomly generated by using randint() function of Matlab. (Function "randint" generates a random scalar that is either zero or one, with equal probability.) The algorithm described above was applied to various images, including those in the USC SIPI image database [27], CorelDraw image database and the JPEG2000 test images.

For all of the test images the proposed algorithm works well. Figure 34 displays a marked medical image and a marked JPEG2000 test image (N1A) generated using the proposed algorithm. Compared with Figure 33(b) and Figure 4(b), it is obvious that the salt-and-pepper noise does not exist with the proposed algorithm (while the similar amount of data was embedded), and the visual quality of stego-images produced by the proposed algorithm is much higher. Specifically, the PSNR of the marked medical image with respect to the original is 38.53dB, as opposed to the PSNR of 9.28dB shown in Figure 33(b). For the N1A JPEG2000 test image, the PSNR is 42.67dB, instead of 18.01dB as shown in Figure 4(b).



(a) Original medical image (b) Marked medical image

Figure 33. Original medical image vs. stego-image created using De Vleeschouwer et al.'s algorithm, which exhibits severe salt-and-pepper noise (750 information bits are embedded in the 512x512 image). The PSNR of the stego-image vs. the original image is 9.28 dB.



(a) Marked medical image (b) Marked N1A

Figure 34. Marked images with no salt-and-pepper noise. (For (a) 750 information bits are embedded in the 512×512 image. The PSNR is 38.53dB. For (b) 697 information bits are embedded in the 1536×1920 image. The PSNR is 42.67dB.)

To further analyze the performance of the proposed algorithm, the testing results using the “Lena” image (the most commonly used image in the image processing community) are now presented. “Lena” is an 8-bit grayscale image with a size of 512×512.

1) Data embedding capacity

As discussed previously, in the proposed algorithm the capacity is determined by the block size B. The smaller B is, the larger the number of blocks will be and consequently more bits that can be embedded. Table 10 gives the relationship between the block size and capacity. Error correction coding is used to enhance the robustness of the algorithm, and there are various options available. The stronger the error correction capability is, the smaller the embedding capacity. BCH (15,11) is used in this experimental investigation (For a more complete discussion on ECC use, see Section 2). The capacity can thus calculated as

$$C_{capacity} = \frac{M \times N}{4 \times B \times B} \times \frac{11}{15} \quad (8)$$

where M, N are the size of the image and B is the block size. In Table 2, “Capacity with ECC” denotes the capacity determined with Formula (8), which is sometimes referred to as pure payload, while “Capacity without ECC” indicates the capacity if the ECC is not used.

Table 10. Block size vs. capacity (Lena:512×512×8).

Block size	ECC Scheme	Capacity with ECC (bits)	Capacity without ECC (bits)
5	(15,11)	1907	2601
6	(15,11)	1293	1764
7	(15,11)	950	1296
8	(15,11)	750	1024
9	(15,11)	574	784
10	(15,11)	458	625
11	(15,11)	387	529
12	(15,11)	323	441

2) Visual quality of marked images

In this experiment, the visual quality of the stego-images with respect to the original image is measured with peak signal to noise ratio (PSNR).

From Table 10, it is obvious that smaller block sizes provide larger capacities. However, the maximum absolute mean value of an IWT coefficient in a block m_{max}, which had been defined in Formula (2), will be larger. Therefore a bigger shift quantity S is needed, thus degrading the quality of the marked image causing lower PSNR. Table 11 provides some information to verify this observation. The “Minimum shift value” is defined as the smallest integer that is greater than m_{max}.

Table 11. Effect of block size over block mean.

Block size	m_{max} in HL_1	m_{max} in LH_1	Minimum shift values	PSNR (dB)
5	16.56	7.78	8	40.09
6	10.25	4.56	6	41.87
7	9.64	2.72	4	44.81
8	6.14	2.64	4	44.36
9	4.31	2.63	4	44.18
10	4.00	1.84	2	49.86
11	3.41	1.69	2	49.62
12	3.18	1.55	2	49.46

Figure 35 displays the marked Lena images with minimum shift value in different block sizes. Under scrutiny (for instance, zooming in to 200% and above), it can be clearly observed that the visual quality of the marked image in (a) is worse than that of (b). The PSNR of (b) is 9 dB greater than that of (a).

(a) B=5, S=8, PSNR=40.09dB (b) B=12, S=2, PSNR=49.46dB

Figure 35. Image quality of different block size with minimum shift value.

Under the same block size, different shift values can be applied. Later, it will be demonstrated that the larger the shift value, the more robustness the stego-image can obtain. Table 12 shows the PSNR of stego-images using different block sizes and different shift values. It is evident from the table that the price for larger shift value is lower PSNR value, i.e. lower image quality.

Table 12 Results of PSNR under various block size and shift value.

PSNR(dB)		Block Size							
		5	6	7	8	9	10	11	12
Shift Value	2	-	-	-	-	-	49.86	49.62	49.46
	4	-	-	44.81	44.36	44.18	44.08	43.80	43.63
	6	-	41.87	41.30	40.84	40.67	40.55	40.28	40.12
	8	40.09	39.44	38.85	38.39	38.20	38.10	37.83	37.66
	10	38.14	37.48	36.89	36.43	36.26	36.15	35.88	35.71
	12	36.57	35.91	35.32	34.86	34.68	34.57	34.30	34.14
	14	35.23	34.57	33.98	33.52	33.34	33.23	32.96	32.79
	16	34.08	33.42	32.83	32.37	32.18	32.08	31.81	31.64
	18	33.05	32.38	31.80	31.34	31.16	31.05	30.78	30.61
	20	32.13	31.47	30.89	30.43	30.24	30.15	29.86	29.70

3) Robustness of hidden data

Different from most lossless watermarking algorithms, the proposed IWT-based lossless watermarking algorithm is robust to incidental alterations. In particular, as previously mentioned this algorithm is robust to JPEG2000 compression. In the experimental study, the marked images are JPEG2000 lossy compressed with increasing compression ratios. The robustness against JPEG2000 compression is measured by data survival rate, meaning that when the resultant data rate after compression is above or equal to the data survival rate the hidden data can be reliably extracted without error. In other words, the lower the data survival rate is, the more robust the stego-image is.

Figure 36 shows the relationship between stego-image PSNR and data survival rate under different block sizes. It can be observed that, in general, the lower the PSNR the lower the survival rate is, and thus the more robust the stego-image is. For the same survival rate, generally the larger the block size the higher the PSNR, and therefore the better the image quality at the expense of a lower capacity. It is obvious that

the image quality, robustness and capacity interact with each other. It is not possible to achieve the best performance among all of these three aspects at the same time. Compromise needs to be made for different purposes for different applications.

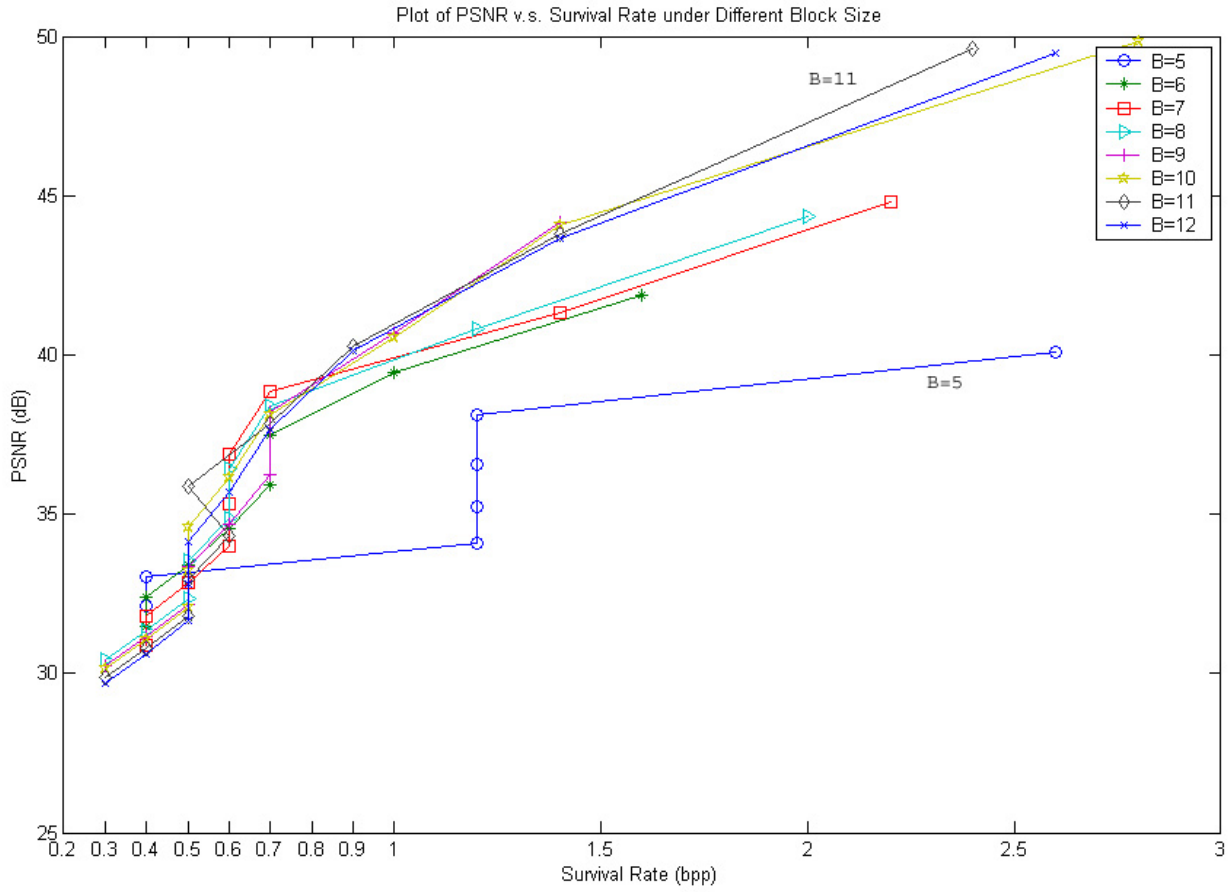


Figure 36. Survival rate vs. PSNR under different block sizes.

In addition to compression, the proposed watermarking scheme is robust to median filtering as well. In these experiments, a median filter with the size of 3 x 3 is applied to the stego-images. Hidden data extraction is then performed on the filtered stego-images. Table 13 lists the test results in which the number of decoded error bits is measured in data extraction procedure. It is obvious that a larger block size and larger shift value will make the marked image more robust against the median filter.

Table 13. Test results against 3×3 median filter.

Decoded error bits		Block Size							
		5	6	7	8	9	10	11	12
Shift Value	2	-	-	-	-	-	228	197	164
	4	-	-	477	383	291	228	197	164
	6	-	643	474	381	275	167	154	131
	8	940	643	359	262	203	91	70	78
	10	940	616	237	160	136	50	37	10
	12	940	485	129	89	79	13	7	0
	14	940	398	82	32	35	0	0	0
	16	940	269	40	14	16	0	0	0
	18	938	199	28	2	11	0	0	0
	20	747	128	11	1	10	0	0	0

Additive Gaussian noise is the noise most frequently encountered in communication systems. In these experiments, a Gaussian noise with zero mean and variance of 0.001 is added to the marked images. The error bits in hidden data extraction are counted and Table 14 lists the results. The similar conclusion can be made that the larger the block size and shift value, the more robust the stego-image will be.

Table 14. Test results against additive Gaussian noise with zero mean and variance of 0.001.

Decoded error bits		Block Size							
		5	6	7	8	9	10	11	12
Shift Value	2	-	-	-	-	-	140	137	121
	4	-	-	200	120	46	27	10	0
	6	-	313	111	32	8	4	0	0
	8	657	236	87	18	3	0	0	0
	10	312	159	19	9	0	0	0	0
	12	46	28	4	2	0	0	0	0
	14	14	6	0	0	0	0	0	0
	16	0	0	0	0	0	0	0	0
	18	0	0	0	0	0	0	0	0
	20	0	0	0	0	0	0	0	0

3.5 Conclusion and Discussion

A new semi-fragile lossless data hiding scheme has been proposed where data is embedded into the integer wavelet domain. As it utilizes IWT it can be integrated into JPEG2000 standard seamlessly. The issue of overflow and underflow is addressed, allowing the original cover media to be obtained after hidden data extraction. The hidden data is also robust to non-malicious attacks (e.g. lossy compression) to a certain degree. Performance analysis has been conducted which shows the advantages of the proposed algorithm. The visual quality of the marked image is excellent. No salt-and-pepper noise or other artifacts are presented in the stego-images. This algorithm can be applied to content-based image authentication and other applications.

4. Conclusion

The research conducted in this project has lead to the following conclusions. First, the problem with the only existing robust lossless data hiding has been identified and analyzed. It has been shown that utilization of modulo-256 addition to achieve losslessness results in salt-and-pepper noise in the marked images. This noise is not acceptable for many applications, such as in medical imaging and remote sensing.

Second, the proposed new robust lossless data hiding algorithm implemented in spatial domain identifies a statistical quantity which is robust to JPEG compression and other incidental alterations and uses it to embed data. Four different types of image block content have been identified, and different embedding strategies are used for different types of blocks. Consequently, there is no salt-and-pepper noise in the marked images. Extensive experimental results have shown that the performance of this new algorithm in terms of data embedding capacity versus visual quality of marked images, and robustness against image compression is better than the prior-art.

Third, a new robust lossless data hiding algorithm implemented in integer wavelet transform (IWT) domain has developed under this project. It applies the above-mentioned embedding strategy in the IWT domain, applying a different embedding scheme for a different type of blocks. Consequently, there is no salt-and-pepper noise in the marked image, and this method can be integrated into the JPEG2000 standard smoothly. The methods described in this report have been included in a proposal entitled “Unified Authentication Framework for JPEG2000 Images” submitted to the Security Part of JPEG2000 standard (known as JPSEC) and is currently included in the JPSEC FCD (final Committee Draft) version 1.0.

References

[1] I. J. Cox, J. Kilian, T. Leighton and T. Shamoon,, "Secure spread spectrum watermarking for multimedia," *IEEE Trans. on Image Processing*, 6, 12, 1673-1687, (1997).

[2] A. Piva, M. Barni, E Bartolini, V. Cappellini, "DCT-based watermark recovering without resorting to the uncorrupted original image", *Proc. ICIP 97*, vol. 1, pp.520

[3] J. Huang and Y. Q. Shi, "An adaptive image watermarking scheme based on visual masking," *IEE Electronic Letters*, vol. 34, no. 8, pp. 748-750, April 1998.

[4] R. Schyndel, A. Tirkel and C. F. Osborne, "A digital watermark," *IEEE International Conference on Image Processing*, pp.86 – 90, vol.2, 1994

[5] B. Chen and G. W. Wornell, "Quantization index modulation: a class of provably good methods for digital watermarking and information embedding," *IEEE Transactions on Information Theory*, vol.47, issue 4, pp.1423 – 1443, May 2001.

[6] Y. Q. Shi, Z. Ni, D. Zou, C. Liang and G. Xuan, "Lossless data hiding: Fundamentals, algorithms and applications," *Proceedings of IEEE International Symposium on Circuits and Systems*, vol. II, pp. 33-36, Vancouver, Canada, May 2004.

[7] C. W. Honsinger, P. Jones, M. Rabbani, and J. C. Stoffel, "Lossless recovery of an original image containing embedded data," US Patent: 6,278,791, 2001.

[8] J. Fridrich, M. Goljan and R. Du, "Invertible authentication," *Proc. SPIE Photonics West, Security and Watermarking of Multimedia Contents III*, Vol. 397, pp. 197-208, San Jose, California, January 2001.

[9] M. Goljan, J. Fridrich, and R. Du, "Distortion-free data embedding," *Proceedings of 4th Information Hiding Workshop*, pp. 27-41, Pittsburgh, PA, April 2001.

[10] G. Xuan, J. Zhu, J. Chen, Y. Q. Shi, Z. Ni, W. Su "Distortionless data hiding based on integer wavelet transform," *Proceedings of IEEE International Workshop on Multimedia Signal Processing*, St. Thomas, US Virgin islands, December 2002. *IEE Electronics Letters*, vol. 38, no. 25, pp.1646-1648, Dec.2002

[11] Z. Ni, Y. Q. Shi, N. Ansari and W. Su, "Reversible Data Hiding," *Proceedings of IEEE International Symposium on Circuits and Systems*, Bangkok, Thailand, May 2003.

[12] M. Celik, G. Sharma, A.M. Tekalp, E. Saber, "Reversible data hiding," in *Proceedings of the International Conference on Image Processing 2002*, pp. 157-160, Rochester, NY, September 2002.

[13] J. Tian, "Reversible data embedding using a difference expansion," *IEEE Transaction on Circuits and Systems for Video Technology*, vol. 13, no. 8, pp. 890-896, August 2003.

[14] C. De Vleeschouwer, J. F. Delaigle and B. Macq, "Circular interpretation of bijective transformations in lossless watermarking for media asset management," *IEEE Tran. Multimedia*, vol. 5, pp. 97-105, March 2003.

[15] W. Bender, D. Gruhl, N. Morimoto and A. Lu, "Techniques for data hiding," *IBM Systems Journal*, vol. 35, no. 3-4, pp. 313-336, 1996.

[16] J. G. Proakis, *Digital communication*, 4th edition, McGraw-Hill 2000.

[17] S. B. Wicker, *Error Control System for Digital Communication and Storage*, Englewood Cliffs, NJ: Prentice-Hall, 1995.

[18] G. Voyatzis and I. Pitas, "Chaotic mixing of digital images and applications to watermarking," *Proceedings of European Conference of Multimedia Applications, Services Techniques (ECMAST'96*), pp.687-695, May 1996.

[19] Z. Ni, Y. Q. Shi, N. Ansari, W. Su, Q. Sun and X. Lin, "Robust lossless image data hiding," *Proceedings of IEEE International Conference on Multimedia and Expo*, Taipei, Taiwan, June 2004.

[20] Z. Zhang, Q. Sun, X. Lin, Y. Q. Shi and Z. Ni, "A unified authentication framework for JPEG2000 images," *Proceedings of IEEE International Conference on Multimedia and Expo*, Taipei, Taiwan, June 2004.

[21] Information Technology – JPSEC (Security Part of JPEG2000) Final Committee Draft (FCD) version 1.0, ISO/IEC JTC 1/SC29/WG1 N3480, November 2004.

[22] A. Skodras, C. Christopoulos, T. Ebrahimi, "The JPEG2000 still image compression standard;" *IEEE Signal Processing Magazine*, vol.18, issue.5, pp36-58, September 2001.

[23] C. Christopoulos, A. Skodras, T. Ebrahimi, "The JPEG2000 still image coding system: an overview," *IEEE Trans. On Consumer Electronics*, vol.46, no.4, pp.1103-1127, Nov. 2000.

[24] JPEG 2000 Part I Final Committee Draft Version 1.0. Available: http://www.jpeg.org/public/fcd15444-1.pdf .

[25] A. R. Calderbank, I. Daubechies, W. Sweldens, and B.-L. Yeo, "Wavelet transforms that map integers to integers," *Applied and Computational Harmonic Analysis (ACHA)*, vol. 5, pp. 332-369, 1998.

[26] L. Gall and A. Tabatabai, "Subband coding of digital images using symmetric short kernel filters and arithmetic coding techniques", *Proc. of IEEE International Conference on Acoustics, Speech and Signal Processing*, NY, pp.761-765, 1988.

[27] USC image database. Avalable: http://sipi.usc.edu/services/database/Database.html .